Apple II

Benutzer Handbuch



HINWEIS

10260 Bandley Drive
Cupertino, CA 95014
U.S.A.

Apple Computer International
7, rue de Chartres
92200 Neuilly-sur-Seine
FRANCE

Apple A2L0001A (030-0004-01) - D

Geschrieben von Christopher Espinosa.

Ins Deutsche übertragen von Albrecht Mengel.

Benutzer Handbuch

EIN NACHSCHLAGEWERK FÜR DIE APPLE II UND APPLE II PLUS COMPUTERSYSTEME

Vorwort

Bevor Sie sich mit diesem Benutzer- Handbuch für die Apple II und Apple II Plus Mikrocomputer genauer befassen, wollen wir Ihnen dazu einige Erläuterungen geben.

Ziel dieses Buches ist es nicht, Ihnen Programmierkenntnisse zu vermitteln, sondern Sie mit einigen wichtigen Grundlagen über Aufbau und Wirkungsweise Ihres Apple vertraut zu machen.

Wenn Sie Ihren Apple gerade erst ausgepackt haben und Sie noch nicht in einer der verfügbaren Sprachen programmieren können, sollten Sie dieses Buch zunächst beiseite legen und erst einmal eines der anderen Anleitungsbücher zu Rate ziehen, die Sie zusammen mit Ihrem Apple erhalten haben. Je nach der Ausführung Ihres speziellen Apple haben sie eines der folgenden Handbücher bekommen :

Apple II BASIC Programmierhandbuch bzw.
Apple II BASIC Programming Manual
(Bestellnummer A2L0005)

Applesoft Progammieranleitung bzw.
The Applesoft Tutorial
(Bestellnummer A2L0018)

Außer der genauen Beschreibung der speziellen BASIC-Sprache Ihres Apple gibt Ihnen das betreffende Handbuch genaue Anleitungen zur Bedienung Ihres Apple. Die Literaturangaben am Ende dieses Buches stellen Ihnen einige weitere interessante Bücher vor.

Dieses Handbuch behandelt alle möglichen unterschiedlichen Ausführungen des Apple. Daher betreffen vielleicht nicht alle in diesem Buch behandelten Möglichkeiten Ihren Apple. An Stellen, an denen dieses Handbuch Modifikationen erwähnt, die nicht bei jedem Apple-Typ vorhanden sind, weisen wir Sie in einer Fußnote auf diese Unterschiede hin.

Wir beschrieben Ihnen den Apple II-Computer mit seinen Einzelteilen und deren Arbeitsweisen. Die einzelnen Kapitel des Handbuches befassen sich mit den System-Monitor, den Ein-/Ausgabegeräten und ihren Funktionen, der internen Organisation des Speichers und der Beschreibung der im Apple verwendeten Elektronik. Weitere Informationen über sonstige Apple Hard- und Software-Produkte finden Sie in entsprechenden Produktbeschreibungen.

Inhaltsverzeichnis

KAPITEL 1
Erste Annäherung an den Apple

KAPITEL 2
Der Dialog mit dem Apple

KAPITEL 3
Der System-Monitor

KAPITEL 4
Der Speicher-Aufbau

KAPITEL 5
Die Organisation der Ein- und Ausgabe

KAPITEL 6
Die Hardware des Apple

ANHANG A
Der Befehlssatz des 6502-Mikroprozessors 119

ANHANG B
Spezial-Adressen 133

ANHANG C
ROM-Programme

GLOSSAR 207

LITERATURHINWEISE 215

INDEX

KAPITEL 1
Erste Annäherung an den Apple

Für detailliertere Informationen, Ihren Apple in Betrieb zu setzen, sei auf Kapitel 1 des APPLE BASIC PROGRAMMIER-HANDBUCHs oder der APPLESOFT PROGRAMMIERANLEITUNG hingewiesen.

In diesem Handbuch halten sich alle Richtungsangaben an diese Orientierung: Sie sehen auf das Schreibmaschine nähnliche Tastenfeld des Apple. „Vorne" und „unten" sind zur Tastatur hin, „hinten" und „oben" sind weg vom Tastenfeld.

Entfernen Sie den Deckel, indem Sie die hintere Ecke hochziehen, bis er ausrastet; dann ziehen Sie den Deckel gerade zurück und heben ihn ab.

Dies hier werden Sie sehen:

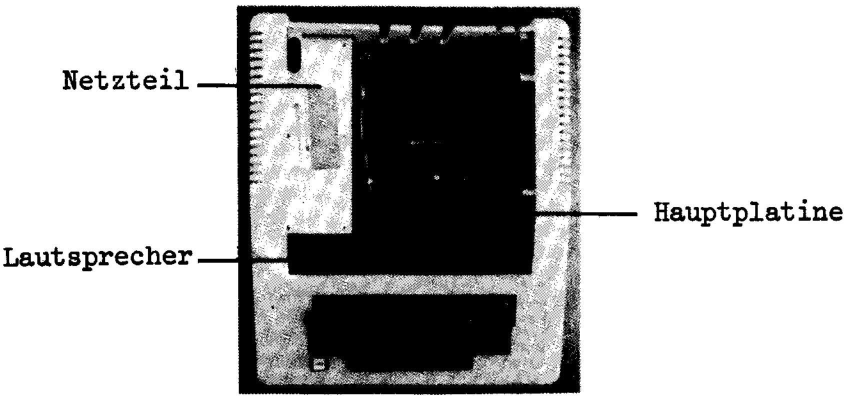
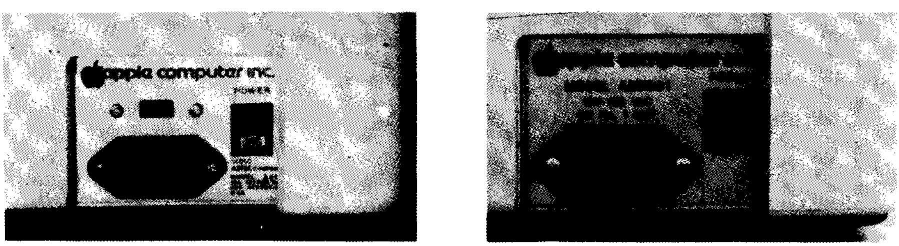


Photo 1. Der Apple II

Die Stromversorgung

Das Metall-Gehäuse innen auf der linken Seite ist das Netzteil. Es liefert vier Spannungen: +5 V, –5,2 V, +11,8 V und –12,0 V. Es ist ein hochfrequent schaltendes Netzgerät mit vielen Schutzeinrichtungen, damit kein Ungleichgewicht zwischen den unterschiedlichen Versorgungen auftritt. Das Netzkabel für den Computer ist direkt an der Hinterseite des Netzgerätes einzustecken. Der Netzschalter befindet sich auch am Netzteil, um sie davor zu schützen, unbeabsichtigt mit der Hochspannungs-Netzversorgung in Berührung zu kommen.



110/220 Volt   110 Volt

Photo 2. Die Rückseite des Apple-Netzteils

Die Hauptplatine

Die große grüne Leiterplatte, die fast das ganze untere Gehäuse ausfüllt, ist der eingentliche Computer. Es gibt zwei leicht unterschiedliche Ausführungen der Apple II Hauptplatine : die Original-Version (Version 0) und die revidierte Version (Version 1). Die leichten Unterschiede zwischen den beiden Versionen liegen in der Elektronik auf der Platine. Diese Unterschiede werden überall in diesem Buch diskutiert. Eine Zusammenfassung der Unterschiede erscheint im Abschnitt „Varianten des Apple" auf Seite 27

Auf der Hauptplatine sind etwa achtzig integrierte Schaltkreise und eine Handvoll anderer Bauteile untergebracht. Im hinteren Zentrum der Platine, direkt vor den acht Einsteckschlitzen mit den goldenen Kontakten, befindet sich ein integrierter Schaltkreis, der größer als alle anderen ist. Dies ist das Gehirn des Apple. Es ist ein 6502-Mikroprozessor in Synertek/MOS Technik. Im Apple läuft er mit 1 023 000 Arbeitstakten pro Sekunde und kann in dieser Zeit über fünfhunderttausend Additionen und Subtraktionen ausführen. Er hat einen Adreßbereich von 65 536 Bytes zu je 8 Bit. Sein Repertoire umfaßt 56 Befehle mit 13 Adressierarten. Dieser Mikroprozessor und andere Versionen werden in vielen Computersystemen gebraucht, genau so wie alle anderen elektronischen Bauteile.

Gleich unter dem Mikroprozessor sind sechs Sockel angeordnet, in denen ein bis sechs etwas kleinere integrierte Schaltkreise stecken. Diese IC's sind die Lesespeicherbausteine, die sogenannten „ROM's". Sie enthalten Programme für den Apple, die im Augenblick des Anschaltens verfügbar werden. Viele Programme sind in ROM's vorhanden. Dazu gehören der Apple System-Monitor, der Apple Autostart-Monitor, Apple Integer-Basic (Ganzzahl-Basic), Applesoft II Basic (Fließkomma-Basic) und das Apple Programmer's Aid#1 (Programmierhilfe Nr. 1) Unterprogrammpaket. Die Anzahl und der Inhalt der ROM's Ihres Apple hängt davon ab, welche Version des Apple und welches Zubehör Sie erworben haben.

Direkt unter den ROM's und der zentralen Befestigungsschraube ist eine Fläche, die durch ein weißes Quadrat eingegrenzt ist. Hier stehen 24 Sockel, die alle oder wenigstens zum Teil mit IC's bestückt sind. Dies sind die „RAM"-Speicherbausteine, die den Hauptspeicher des Apple bilden. Der Apple kann in den drei Reihen 4 096 bis 49 152 Bytes RAM-Speicher aufnehmen*. In jede Reihe passen 8 IC's der Größe 4K oder 16K.

Jede Reihe muß aus 8 Speicherbausteinen desselben Typs bestehen, aber die 2 Typen können auf mehrere Arten in den drei Reihen eingesetzt werden. So kann man neun verschiedene Speichergrößen realisieren**. Der RAM-Speicher wird benötigt, um alle Programme und Daten zu speichern, die Sie zu einem bestimmten Zeitpunkt brauchen. Die in RAM gespeicherten Informationen gehen verloren, wenn Sie den Strom abschalten.

---

* Sie können Ihren RAM-Speicher auf 64K ausdehnen, wenn Sie die Apple Language Card (Apple Sprachkarte), Teil des Apple Language Systems, (Teil Nr. A2B0006) beziehen.

** Der Apple II ist so entworfen, daß er sowohl die 16K- als auch die weniger kostspieligen 4K-RAM's gebrauchen kann. Aber infolge der größeren Brauchbarkeit und der verminderten Kosten für 16K-Bausteine wird der Apple nur mit 16K-RAM's ausgerüstet.

Die anderen Bauteile auf der Apple II-Platine haben unterschiedliche Funktionen: sie kontrollieren im Computer den Informationsfluß von einem Teil zum anderen, sammeln Daten von der Außenwelt oder senden Ihnen Informationen auf den Bildschirm oder als Geräusche auf den Lautsprecher.

Jeder der acht Einsteckschlitze hinten auf der Apple-Platine kann eine Peripheriekarte aufnehmen. So ist es Ihnen möglich, Ihren RAM- oder ROM-Speicherplatz auszudehnen oder einen Drucker oder andere Ein/-Ausgabe-Geräte anzuschließen.

So sprechen Sie zu Ihrem Apple

Die Verbindung zu Ihrem Apple liegt in Ihren Fingerspitzen. Die meisten Programme und Sprachen auf dem Apple erwarten, daß Sie durch das Tastenfeld zu ihnen sprechen. Die Tastatur entspricht einer normalen Schreibmaschinentastatur, außer einigen geringfügigen Umordnungen und ein paar Spezialtasten (Für einen schnellen Überblick über die Tastatur: siehe Seite 4 bis 10 im APPLE II BASIC-PROGRAMMIER-HANDBUCH oder die Seiten 5 bis 11 der APPLESOFT PROGRAMMIEREINFÜHRUNG).

So wie Sie mit Ihren Fingern „sprechen", können Sie auch mit Ihren Augen „hören". Der Apple wird Ihnen sagen, was er tut, indem er Buchstaben, Zahlen, Zeichen und manchmal farbige Blöcke und Linien auf den Bildschirm des Schwarz-weiß- oder Farbfernsehers bringt.

Die Tastatur

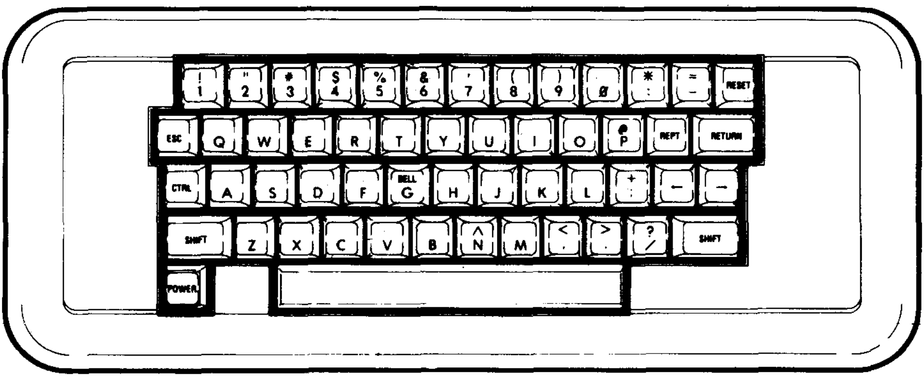
Das Apple Tastenfeld

Anzahl der Tasten	: 52			
Codierung	: ASCII mit Großbuchstaben			
Anzahl der Tastencodes	: 91			
Ausgang	: 7 Bit und Tastimpuls (Strobe)			
Nötige Betriebsspannungen	: +5V bei 120 mA			
	: -12V bei 50 mA			
Spezialtasten	: CTRL			
	ESC			
	RESET			
	REPT			
Zugeordnete Speicherzellen:		Hexa	Dezimal	
	Inhalt :	$CØØØ	49152	-16384
	Löschen :	$CØ1Ø	49168	-16368

Der Apple II hat eine eingebaute schreibmaschinenähnliche Tastatur mit 52 Tasten, die im ASCII-Code* arbeiten. 91 der 96 ASCII-Zeichen mit Großbuchstaben können direkt vom Tastenfeld generiert werden. Tabelle 2 zeigt die Tasten der Tastatur und ihre zugeordneten ASCII-Codes. „Photo" 3 ist ein Diagram des Tastenfeldes.

Das Tastenfeld ist durch ein 16-adriges Kabel mit der Hauptplatine verbunden. Die Stecker des Kabels befinden sich in Standard-IC-Sockeln an der Tastatur und ganz vorne an der Hauptplatine, direkt unter dem Tastenfeld. Die elektrischen Spezifikationen für diesen Anschluß finden Sie auf Seite 106

Die meisten Sprachen auf dem Apple haben Kommandos oder Anweisungen, die Ihrem Programm erlauben, Eingaben von der Tastatur schnell und einfach entgegenzunehmen. (z.B. die INPUT- und GET-Anweisungen in Basic). Jedoch kann Ihr Programm auch direkt das Tastenfeld lesen.



„Photo" 3. Das Apple Tastenfeld

Funktion der Tastatur

Das Tastenfeld gibt 7 Bits aus, die zusammen ein Zeichen bilden. Diese und ein anderes Signal, das anzeigt, ob eine Taste gedrückt ist, sind den meisten Programmen als Inhalt einer bestimmten Speicherstelle verfügbar.

Programme können den aktuellen Zustand des Tastenfeldes abfragen, indem sie den Wert dieser Speicherstelle lesen. Dieser Wert wird 128 oder größer, wenn Sie eine Taste drücken und stellt genau den numerischen Code des Zeichens dar, dessen Taste gerade betätigt wurde. Tabelle 3 auf Seite 8 zeigt die ASCII-Zeichen und die zugeordneten numerischen Codes. Die Speicherstelle behält ihren Wert, bis Sie eine andere Taste drücken oder bis Ihr Programm der Speicherstelle anweist, das Zeichen zu vergessen, das es gerade enthält.

* Alle ASCII-Codes haben beim Apple normalerweise das führende Bit gesetzt.

Hat Ihr Programm einmal einen Tastendruck akzeptiert und verstanden, sollte es die Speicherstelle dazu bringen, das gehaltenen Zeichen zu „entlassen" und auf ein neues zu warten. Ihr Programm kann dies mit dem Zugriff auf eine andere Speicherstelle erreichen. Wenn Sie diese andere Speicherstelle ansprechen, wird der Wert in der ersteren Speicherstelle unter 128 fallen. Dieser Wert bleibt so, bis Sie eine andere Taste betätigen. Dieser Vorgang heißt „Tastenimpuls löschen" (Clear Keyboard Strobe). Ob Ihr Programm nun beim Zugriff auf diese Speicherstelle liest oder schreibt, der erhaltene oder geschriebene Wert ist unwichtig. Es kommt allein auf den Zugriff auf diese Speicherstelle an, um den Tastenimpuls zu löschen. Ist der Tastenimpuls einmal gelöscht, können Sie durch Addition von 128 (hexadezimal $8Ø) zu dem Wert des Tastendruckspeichers den Code der zuletzt gedrückten Taste wiederherstellen.

Dies sind die speziellen Speicherstellen für die Tastatur:

Tabelle 1: Adressen des Tastenfeldes

Adresse			Beschreibung
Hexa	Dezimal		
$CØØØ	49152	-16384	Tastendruckspeicher
$CØ1Ø	49168	-16368	Tastendruck löschen

Die RESET-Taste rechts oben generiert keinen ASCII-Code, sie ist direkt mit dem Mikroprozessor verbunden. Wenn diese Taste gedrückt wird, hört jeder Prozess auf. Beim Loslassen startet der Computer den RESET-Ablauf (die Beschreibung der RESET-Funktion findet sich auf Seite 39).

Die CTRL- und SHIFT-Tasten generieren selbst kein Zeichen, verändern dafür aber die Codes anderer Tasten.

Die REPT-Taste erzeugt ein Duplikat des letzten generierten Codes, falls sie allein betätigt wird. Halten Sie die REPT-Taste gedrückt, während Sie eine andere Taste betätigen, so wird alle zehntel Sekunde ein Tastendruck dieser Taste produziert. Diese dauernde Wiederholung stoppt, sobald Sie eine der beiden Tasten loslassen.

Die POWER-Leuchte links unten zeigt an, daß der Apple eingeschaltet ist. Es handelt sich nicht um den Betriebsschalter.

TABELLE 2 : Tasten und die zugehörigen ASCII-Codes.

Taste	Allein	CTRL	SHIFT	Beide	Taste	Allein	CTRL	SHIFT	Beide
Leertaste	$AØ	$AØ	$AØ	$AØ	RETURN	$8D	$8D	$8D	$8D
Ø	$BØ	$BØ	$BØ	$BØ	G	$C7	$87	$C7	$87
1!	$B1	$B1	$A1	$A1	H	$C8	$88	$C8	$88
2"	$B2	$B2	$A2	$A2	I	$C9	$89	$C9	$89
3#	$B3	$B3	$A3	$A3	J	$CA	$8A	$CA	$8A
4$	$B4	$B4	$A4	$A4	K	$CB	$8B	$CB	$8B
5%	$B5	$B5	$A5	$A5	L	$CC	$8C	$CC	$8C
6&	$B6	$B6	$A6	$A6	M	$CD	$8D	$DD	$9D
7'	$B7	$B7	$A7	$A7	N^	$CE	$8E	$DE	$9E
8(	$B8	$B8	$A8	$A8	O	$CF	$8F	$CF	$8F
9)	$B9	$B9	$A9	$A9	P@	$DØ	$9Ø	$CØ	$8Ø
:*	$BA	$BA	$AA	$AA	Q	$D1	$91	$D1	$91
;+	$BB	$BB	$AB	$AB	R	$D2	$92	$D2	$92
,<	$AC	$AC	$BC	$BC	S	$D3	$93	$D3	$93
-=	$AD	$AD	$BD	$BD	T	$D4	$94	$D4	$94
.>	$AE	$AE	$BE	$BE	U	$D5	$95	$D5	$95
/?	$AF	$AF	$BF	$BF	V	$D6	$96	$D6	$96
A	$C1	$81	$C1	$81	W	$D7	$97	$D7	$97
B	$C2	$82	$C2	$82	X	$D8	$98	$D8	$98
C	$C3	$83	$C3	$83	Y	$D9	$99	$D9	$99
D	$C4	$84	$C4	$84	Z	$DA	$9A	$DA	$9A
E	$C5	$85	$C5	$85	←	$88	$88	$88	$88
F	$C6	$86	$C6	$86	→	$95	$95	$95	$95
					ESC	$9B	$9B	$9B	$9B

Alle Codes sind hexadezimal. Die Dezimalwerte dazu findet man in Tabelle 3.

TABELLE 3 : Der ASCII Zeichensatz.

Dezimal	Hexa	128 $8ø	144 $9ø	16ø $Aø	176 $Bø	192 $Cø	2ø8 $Dø	224 $Eø	24ø $Fø
ø	$ø	nul	dle		ø		P		p
1	$1	soh	dc1	!	1	A	Q	a	q
2	$2	stx	dc2	"	2	B	R	b	r
3	$3	etx	dc3	#	3	C	S	c	s
4	$4	eot	dc4	$	4	D	T	d	t
5	$5	enq	nak	%	5	E	U	e	u
6	$6	ack	syn	&	6	F	V	f	v
7	$7	bel	etb	'	7	G	W	g	w
8	$8	bs	can	(	8	H	X	h	x
9	$9	ht	em	)	9	I	Y	i	y
1ø	$A	lf	sub	*	:	J	Z	j	z
11	$B	vt	esc	+	;	K	[	k	{
12	$C	ff	fs	,	<	L	\	l	|
13	$D	cr	gs	-	=	M	]	m	}
14	$E	so	rs	.	>	N	^	n	~
15	$F	si	us	/	?	O	_	o	rub

Gruppen von zwei oder drei Kleinbuchstaben sind Abkürzungen für Standard-ASCII Steuerzeichen.

Nicht alle Zeichen in dieser Tabelle können vom Tastenfeld erzeugt werden. Es handelt sich dabei um die Zeichen aus den beiden Spalten, die am weitesten rechts liegen, die Symbole [(eckige Klammer links), \ (rückwärtiger Schrägstrich), _ (unterstreichen) und die Steuerzeichen „fs", „us" und „rub".

Der dezimale oder hexadezimale Wert eines jeden Zeichens aus der Tabelle ergibt sich aus der Summe der Dezimalzahlen oder Hexadezimalzahlen oben in der Spalte und links in der Zeile des betrachteten Zeichens.

Die Apple Bildschirmeinheit

Die optische Ausgabe des Apple

Speicherung des Bildinhaltes	: innerhalb des RAM-Hauptspeichers
Anzeige-Arten	: Text, Block-Graphik, hochauflösende Graphik
Text-Kapazität	: 960 Zeichen (24 Zeilen, 40 Spalten)
Zeichentyp	: 5 x 7 Punktmatrix
Zeichen-Arten	: normal, invers, blinkend
Zeichensatz	: ASCII-Großbuchstaben, 64 Zeichen
Graphik-Kapazität	: Block-Graphik: 1920 Blöcke in 48 Zeilen und 40 Spalten Hochauflösende Graphik: 53760 Punkte in 192 Zeilen und 280 Spalten
Anzahl der Farben	: 16 (Block-Graphik) 6 (hochauflösende Graphik)

Der Video-Anschluss

Hinten rechts auf der Apple II-Platine befindet sich ein metallener Anschluß, der mit „Video" gekennzeichnet ist. An diesen Anschluß können Sie ein Kabel zwischen dem Apple und einem Monitor mit direktem Video-Eingang anbringen. Das eine Ende des Verbindungskabels sollte einen RCA-Phono-Stecker für den Anschluß am Apple haben und das andere Ende einen passenden Anschluß für das Gerät, das Sie benutzen. Das Signal am Videoanschluß des Apple entspricht der EIA-Norm (Electronic Industries Association) und es ist NTSC-kompatibel (National Television Standards Committee) mit positivem Farbvideosignal (Bildaustast-synchronsignal). Der Pegel dieses Signals kann von 0 V bis 1 V Spitzenwert mit einem kleinen runden Potentiometer eingestellt werden, das sich rechts außen auf der Hauptplatine, etwa 5,5 cm vor dem hinteren Ende der Platine befindet.

Eine nicht justierbare 2 V-ausführung desselben Video-Signals steht an zwei anderen Stellen zur Verfügung: ein einzelner Anschlußstift* für „wire-wrap"-Verbindung rechts hinten etwa 1,5 cm über dem Video-Potentiometer und die Vier-Stift Video-Anschlußbuchse etwa 3 cm über dem Video-Potentiometer. Die drei anderen Stifte der Viererbuchse sind mit −5 V, +12 V und mit Masse verbunden (siehe Seite 101 : Video-Hilfsanschluß).

* Dieser Stift ist nicht auf Apple II-Platinen der Version 0 vorhanden.

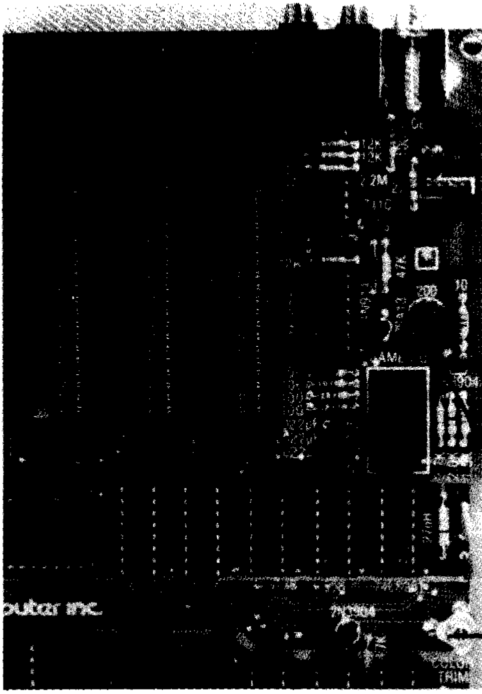


Photo 4. Video-Anschlüsse und Potentiometer

Bildschirmformat

Auf dem an den Apple angeschlossenen Bildschirm können Informationen auf drei verschiedene Weisen sichtbar gemacht werden :

1) **Text.** Der Apple kann 24 Zeilen aus Zahlen, Spezialzeichen und Großbuchstaben auf den Bildschirm bringen. Jede Zeile besteht aus 40 Zeichen, und jedes Zeichen setzt sich aus einer 35-Punkt-Matrix (7 Punkte hoch und 5 Punkte breit) zusammen. Rechts und links und über jedem Zeichen schließt sich ein ein-Punkt-breiter Zwischenraum an.

2) **Block-Graphik (Lo-Res = Low Resolution).** Der Apple kann 1920 farbige Flächen in einem Feld von 40 Blöcken Breite und 48 Blöcken Höhe darstellen. Für jeden Block stehen 16 verschiedene Farben zur Verfügung. Es erscheint kein Zwischenraum zwischen den einzelnen Blöcken, so daß 2 benachbarte Felder wie ein einziger größerer Block erscheinen.

3) **Hochauflösende Graphik (Hi-Res = High Resolution).** Der Apple kann auch farbige Punkte in einem 280 Punkte breiten und 192 Punkte hohen Feld auf dem Bildschirm wiedergeben. Die Punkte haben dieselbe Größe wie die Punkte, aus denen die Textzeichen bestehen. Für die hochauflösende Graphik stehen sechs Farben zur Verfügung: schwarz, weiß, rot, blau, grün und violett*. Jeder Punkt auf dem Bildschirm kann schwarz, weiß oder farbig sein, wenn auch nicht alle Farben für jeden Punkt zur Verfügung stehen.

Wenn der Apple einen bestimmten Informationstyp auf den Bildschirm bringt, befindet er sich in einem bestimmten „Modus". Wenn Sie also Worte und Zahlen auf dem Bildschirm sehen, können Sie ziemlich sicher sein, daß Ihr Apple im Textmodus ist. Entsprechend verhält es sich, wenn Sie den Bildschirm voller gefärbter Blöcke sehen: Ihr Computer ist offensichtlich im Block Graphik-Modus. Sie können auch bei jedem Graphik-Modus einen vierzeiligen „Textbereich" unten auf dem Bild haben.

Diese vier Zeilen erscheinen anstatt der unteren 8 Zeilen von Blöcken der Block-Graphik, wodurch noch ein Feld von 40 mal 40 übrigbleibt. In hochauflösender Graphik werden dann 32 Punktzeilen durch den Text ersetzt, so daß noch eine 160 mal 280 Punktmatrix für die Graphik zur Verfügung bleibt. In jedem „gemischten Modus" wird Text und Graphik gleichzeitig auf den Bildschirm gebracht, aber es gibt keine Möglichkeit, beide Graphik-Arten zur selben Zeit zu bringen.

Bildspeicherung

Der Video-Teil holt sich die nötigen Informationen aus dem RAM-Hauptspeicher, um ein Bild zu generieren. Der Inhalt einer einzigen Speicherstelle entscheidet über das Erscheinen eines bestimmten Objektes auf einer festgelegten Stelle auf dem Bildschirm. Dies Objekt kann ein Zeichen sein, zwei aufeinandergestapelte farbige Blöcke oder eine Zeile aus sieben Punkten. Im Text- und Lo-Res-Graphik-Modus wird ein Speicherbereich von 1024 Bytes als Quelle für die Bildschirminformation benutzt. Text- und Lo-Res-Graphik arbeiten mit diesem Speicherbereich. Bei der Hi-Res-Graphik ist ein anderer größerer Speicherbereich (8192 Bytes) für die größere anzuzeigende Informationsmenge nötig. Diese Speicherbereiche nennt man auch „Seiten". Im Falle der Hi-Res-Graphik spricht man von „Bildpuffer", weil dort gewönhlich Bilder oder Zeichnungen aufbewahrt werden.

Bildschirmseiten

Es gibt sogar zwei Bereiche, in die jeder Modus seine Information schreiben kann. Der erste Bereich wird „erste Seite" oder „Seite 1" genannt, der zweite „Seite 2" oder „zweite Seite". Die beiden Bereiche sind gleich groß und folgen direkt aufeinander. Die zweite Seite ist nützlich, um Bilder oder Texte zu speichern und diese ohne Verzögerung auf den Bildschirm zu bringen. Ein Programm kann Zeichentricks darstellen; indem es in die eine Seite zeichnet, während die andere auf dem Bildschirm sichtbar ist und dann plötzlich die Seiten tauscht.

Text und Block-Graphik teilen sich denselben Bereich für die zweite Seite, genauso wie sie den Speicherbereich für Seite 1 gemeinsam benutzen. Die beiden oben beschriebenen gemischten Anzigearten sind auch auf Seite 2 verfügbar. Es gibt aber keinen Weg, die zwei Seiten auf denselben Bildlchirm zu bringen.

---

* Apples mit der Platinen-Version 0 haben 4 Farben: schwarz, weiß, grün und violett.

Tabelle 4: Bildschirm-Speicherbereiche					
Modus	Seite	Beginn		Ende	
		Hexa	Dezimal		
Text/Lo-Res	1	$400	1024	$7FF	2047
	2	$800	2048	$BFF	3071
Hi-Res	1	$2000	8192	$3FFF	16383
	2	$4000	16384	$5FFF	24575

Bildmodus-Schalter

Die Vorrichtungen, die zwischen den verschiedenen Darstellungsarten, Seiten und gemischten Anzeigen entscheiden, sind „Programmschalter". Es sind Schalter, weil sie zwei Positionen haben (z.B. an oder aus, Text oder Graphik) und es sind „Programm"-Schalter, weil sie vom Programm aus bedient werden.

Ein Programm kann auf einen Schalter durch Zugriff auf eine spezielle Speicherstelle einwirken. Die Daten, die im Zusammenhang mit dieser Speicherstelle gelesen oder geschrieben werden, sind unwichtig: allein der Zugriff auf die Adresse betätigt den Schalter.

Es gibt acht spezielle Speicherstellen, die die Programmschalter für den Bildschirm kontrollieren. Sie sind paarweise und komplementär angeordnet. Beim Zugriff auf die eine Adresse eines Paares stellen Sie den entsprechenden Modus „an" und den entgegengesetzten Modus „aus". Die Paare sind:

Tabelle 5: Bildmodus-Schalter			
Adresse			Beschreibung
Hexa	Dezimal		
$C050	49232	-16304	Anzeige auf GRAPHIK-Modus.
$C051	49233	-16303	Anzeige auf TEXT-Modus.
$C052	49234	-16302	Anzeige auf: nur TEXT oder nur GRAPHIK.
$C053	49235	-16301	Gemischte Anzeige: TEXT und GRAPHIK.*
$C054	49236	-16300	Anzeige der 1. Seite.
$C055	49237	-16299	Anzeige der 2. Seite.
$C056	49238	-16298	Anzeige auf Lo-Res GRAPHIK.*
$C057	49239	-16297	Anzeige auf Hi-Res GRAPHIK.*

* Diese Zustände sind nur sichtbar, wenn der „Graphik-Anzeige"-Schalter auf „an" ist.

Dies sind die zehn unterschiedlichen Kombinationen der Bildmodus-Schalter:

Tabelle 6: Bildmodus-Kombinationen

Seite 1 Anzeige	Schalter		Seite 2 Anzeige	Schalter	
Alles Text	$CØ54	$CØ51	Alles Text	$CØ55	$CØ51
Alles Lo-Res	$CØ54	$CØ56	Alles Lo-Res	$CØ55	$CØ56
Graphik	$CØ52	$CØ5Ø	Graphik	$CØ52	$CØ5Ø
Alles Hi-Res	$CØ54	$CØ57	Alles Hi-Res	$CØ55	$CØ57
Graphik	$CØ52	$CØ5Ø	Graphik	$CØ52	$CØ5Ø
Gemischt Text	$CØ54	$CØ56	Gemischt Text	$CØ55	$CØ56
und Lo-Res	$CØ53	$CØ5Ø	und Lo-Res	$CØ53	$CØ5Ø
Gemischt Text	$CØ54	$CØ57	Gemischt Text	$CØ55	$CØ57
und Hi-Res	$CØ53	$CØ5Ø	und Hi-Res	$CØ53	$CØ5Ø

(Nun werden sich einige von Ihnen, die sich gut im Binären auskennen, wundern: „Wo bleiben die restlichen sechs Möglichkeiten?!", wohl wissend, daß 4 zwei-Weg-Schalter genau 16 Kombinationen ermöglichen. Die Antwort auf das Geheimnis der sechs fehlenden Zustände liegt in dem TEXT/GRAPHIK-Schalter. Befindet sich der Computer im TEXT-Modus, so kann er sich auch in einer der sechs Kombinationen aus Lo-Res-/Hi-Res Graphik, gemischten Zustand oder Seitenauswahl befinden. Da der Apple nun aber Text anzeigt, sind diese unterschiedlichen graphischen Zustände nicht sichtbar.)

Um den Apple in einen dieser Zustände zu setzen, braucht ein Programm nur auf die Adressen der Speicherstellen zuzugreifen, die die Programmschalter für diesen Modus darstellen. Programme in Maschinensprache brauchen dazu die oben gegebenen hexadezimalen Adressen: BASIC-Programme arbeiten mit PEEK und POKE und den entsprechenden Dezimalzahlen (diese sind oben in Tabelle 5 gegeben). Die Schalter können in jeder Reihenfolge betätigt werden, jedoch empfiehlt es sich beim Umschalten in einen der Graphikzustände den TEXT/GRAPHIK-Schalter zuletzt zu bedienen. Alle anderen Zustandswechsel vollziehen sich dann unsichtbar hinter dem Text, so daß beim Setzen des Graphik-Modus die fertige Graphik-Anzeige im ganzen erscheint.

Der Textmodus

Im Textzustand kann der Apple 24 Zeilen mit bis zu 4Ø Zeichen pro Zeile anzeigen. Jedes Zeichen auf dem Bildschirm entspricht dem Inhalt einer Speicherstelle aus dem Speicherbereich der Seite, die zur Anzeige gebraucht wird. Der Zeichensatz besteht aus 26 Großbuchstaben, 10 Ziffern und 28 Spezialzeichen. Die 64 Zeichen werden aus einer 5 Punkte breiten und 7 Punkte hohen Matrix gebildet. An beiden Seiten eines jeden Zeichens gibt es einen ein-Punkt-breiten Zwischenraum, um benachbarte Zeichen zu trennen. Außerdem werden benachbarte Zeilen durch einen ein-Punkt-hohen Abstand auseinandergehalten. Jedes Zeichen ist normalerweise aus weißen Punkten auf schwarzem Untergrund gebildet, kann jedoch auch schwarz punktiert auf weißem Hintergrund erscheinen oder dauernd dazwischen wechseln, und ein blinkendes Zeichen erzeugen. Im Textmodus schaltet der Video-Schaltkreis des Apple das Farbsignal für den Bildmonitor aus, so daß sie ein klares Schwarz-Weiß-Bild haben*.

* Diese Einrichtung gibt es nicht auf der Platine der Version Ø.

Der Speicherbereich für die erste Textseite beginnt ab Adresse 1024 und erstreckt sich bis zur Adresse 2047. Die zweite Seite wird in die Speicherplätze 2048 bis 3071 abgelegt. In Maschinensprache ergeben sich für die erste Seite die Hexadezimaladressen $400 bis $7FF und für die zweite Seite $800 bis $BFF. Jede dieser Seiten ist 1024 Bytes lang.

Wer sich nicht vor der Multiplikation fürchtet, wird feststellen, daß nur 960 Zeichen angezeigt werden. Die verbleibenden 64 Bytes auf jeder Seite, die nicht auf dem Bildschirm erscheinen, werden gebraucht, um Programmen in PROM's auf den Apple „Intelligent Interface"-Gerätekarten als zeitweilige Speicherstellen zu dienen (siehe Seite 87).

Photo 5 zeigt 64 Zeichen, die auf dem Bildschirm des Apple verfügbar sind.

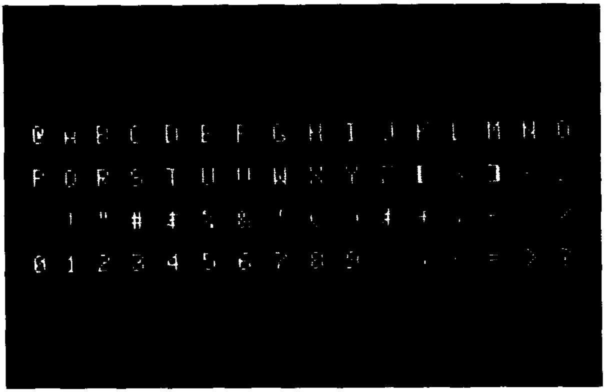


Photo 5. Der Zeichensatz des Apple

Tabelle 7 zeigt die dezimalen und hexadezimalen Codes der 64 Zeichen in normalen, inversem und blinkendem Zustand.

	Invers				Blinkend				CTRL		Normal				Klein buchstaben	
Decimal	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
Hex	$00	$10	$20	$30	$40	$50	$60	$70	$80	$90	$A0	$B0	$C0	$D0	$E0	$F0
0 $0	@	P		0	@	P		0	@	P		0	@	P		0
1 $1	A	Q	!	1	A	Q	!	1	A	Q	!	1	A	Q	!	1
2 $2	B	R	"	2	B	R	"	2	B	R	"	2	B	R	"	2
3 $3	C	S	#	3	C	S	#	3	C	S	#	3	C	S	#	3
4 $4	D	T	$	4	D	T	$	4	D	T	$	4	D	T	$	4
5 $5	E	U	%	5	E	U	%	5	E	U	%	5	E	U	%	5
6 $6	F	V	&	6	F	V	&	6	F	V	&	6	F	V	&	6
7 $7	G	W	'	7	G	W	'	7	G	W	'	7	G	W	'	7
8 $8	H	X	(	8	H	X	(	8	H	X	(	8	H	X	(	8
9 $9	I	Y	)	9	I	Y	)	9	I	Y	)	9	I	Y	)	9
10 $A	J	Z	*	:	J	Z	*	:	J	Z	*	:	J	Z	*	:
11 $B	K	[	+	;	K	[	+	;	K	[	+	;	K	[	+	;
12 $C	L	\	,	<	L	\	,	<	L	\	,	<	L	\	,	<
13 $D	M	]	–	=	M	]	–	=	M	]	–	=	M	]	–	=
14 $E	N	^	.	>	N	^	.	>	N	^	.	>	N	^	.	>
15 $F	O	_	/	?	O	_	/	?	O	_	/	?	O	_	/	?

Tabelle 7: ASCII Bildschirmzeichen

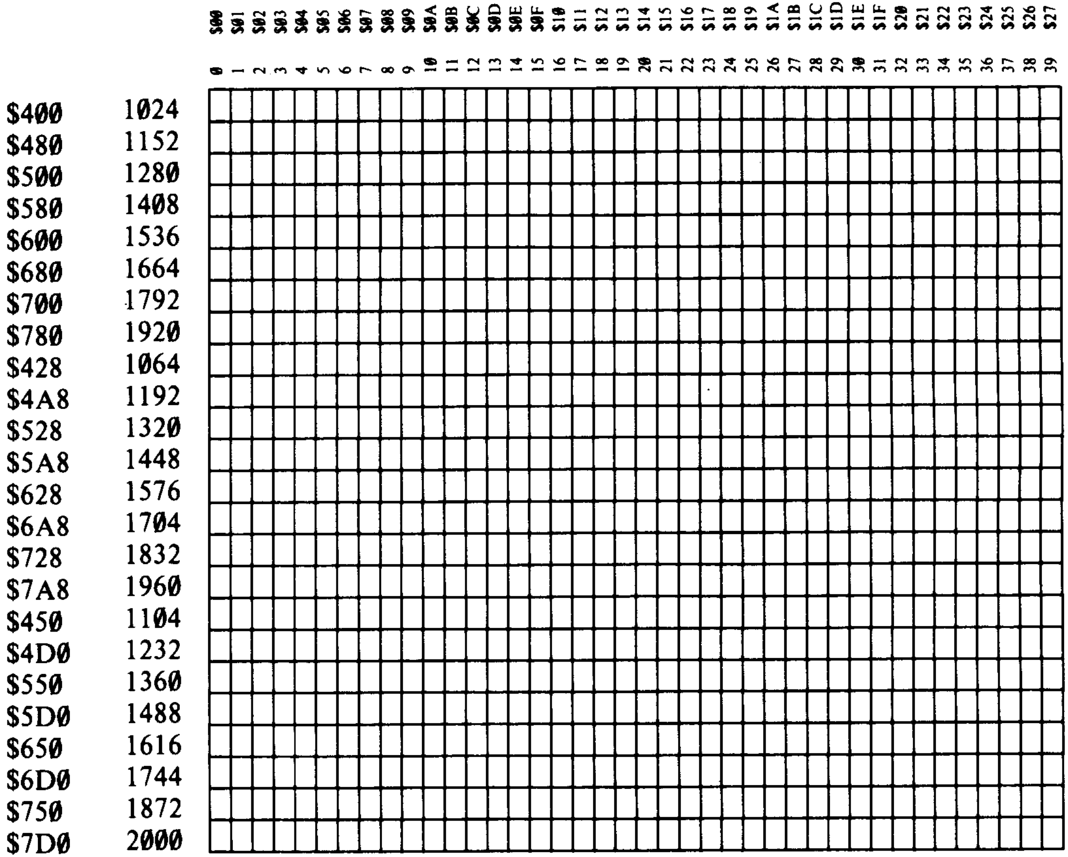


Schaubild 1: Speicherplan des Text-Bildschirms

Schaubild 1 zeigt für den Text-Modus die Speicheradresse jedes Zeichens auf dem Bildschirm des Apple.

Die Block-Graphik

Im Block-Graphik-Zustand präsentiert der Apple dieselben 1024 Speicherstellen wie im Text-Modus, nur in einem anderen Format. In diesem Zustand wird jedes Byte nicht als ASCII-Zeichen, sondern als zwei aufeinandergestellte farbige Blöcke angezeigt. Der Bildschirm kann ein 40 Blöcke hohes Feld aufweisen. Jeder Block hat eine von 16 Farben. Auf einem Schwarz-Weiß-Fernsehgerät erscheinen die Farben als Muster aus grauen und weißen Punkten.

Da jedes Byte in der Speicherseite für die Lo-Res-Graphik zwei übereinanderstehende Blöcke auf dem Bildschirm darstellt, wird es in zwei gleich große Teile zerlegt. (Diese Teilung kennen Sie schon von der Hexadezimaldarstellung : je vier Bits werden zu einer Hexadezimalziffer zusammengefaßt.) Die beiden Hälften können jeweils Werte von 0 bis 15 annehmen. Die linken vier Bits (Bit 7 bis Bit 4) bestimmen die Farbe des oberen Blocks und die rechten vier Bits (Bit 3 bis Bit 0) legen die Farbe des unteren Blocks des auf dem Bildschirm wiederzugebenden Bytes fest. Die Farben sind von 0 bis 15 durchnummeriert :

Tabelle 8: Farben der Block-Graphik

Dezimal	Hex	Farbe	Dezimal	Hex	Farbe
0	$0	Schwarz	8	$8	Braun
1	$1	Fuchsinrot	9	$9	Orange
2	$2	Dunkelblau	10	$A	Grau 2
3	$3	Purpur	11	$B	Rosa
4	$4	Dunkelgrün	12	$C	Hellgrün
5	$5	Grau 1	13	$D	Gelb
6	$6	Blau	14	$E	Aquamarin
7	$7	Hellblau	15	$F	Weiß

(Die Farben können von Fernseher zu Fernseher abweichen, besonders bei denen ohne Farbwertkontrolle. Sie können den Farbton mit dem Farbtrimmer rechts hinten auf der Platine justieren.)

Ein Byte mit dem hexadezimalen Wert $D8 erscheint also als ein brauner Block auf einem gelben Block auf dem Bildschirm. Bei dezimaler Rechenweise ergibt sich die Farbe des unteren Blocks, wenn man den Wert des Bytes durch 16 teilt. Der Rest der Division legt dann die Farbe des oberen Blocks fest.

Skizze 2 zeigt die Speicheradressen aller Blöcke auf dem Bildschirm.

Da die Lo-Res-Graphik denselben Speicherbereich wie die Textausgabe benutzt, passieren interessante Dinge, wenn man zwischen TEXT und Lo-Res-Graphik umschaltet. Ist der Lo-Res-Graphik-Bildschirm voller farbiger Blöcke, so erscheinen nach dem Umschalten des TEXT/GRAPHIK-Programmschalters lauter scheinbar zufällige Textzeichen, die zum Teil invers erscheinen oder blinken. Entsprechend verhält es sich, wenn ein Bildschirm voller Text in den Lo-Res Graphik-Modus geschaltet wird: Lange graue, rosa, grüne oder gelbe waagerechte Streifen werden durch scheinbar zufällig gefärbte Blöcke getrennt.

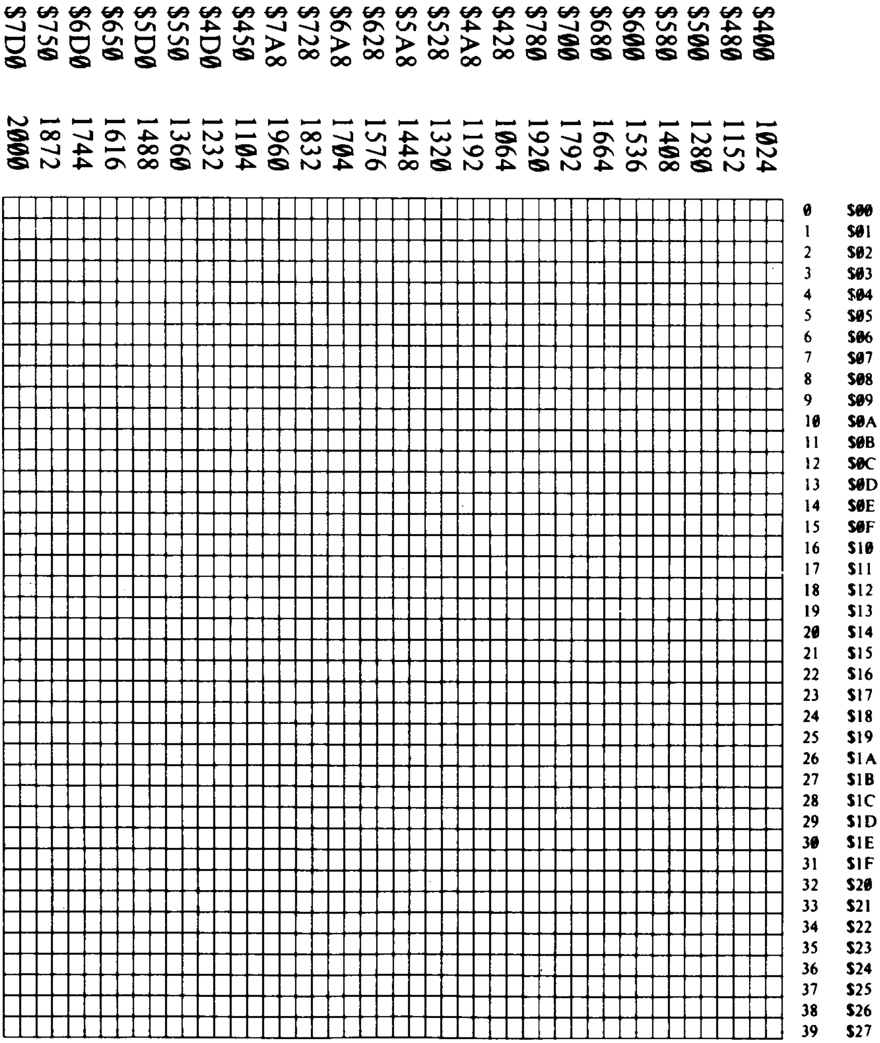


Schaubild 2: Speicherplan der Block-Graphik.

Die hochauflösende Graphik (Hi-Res)

Der Apple hat einen zweiten Typ der Graphik-Anzeige, die hochauflösende Graphik (Abkürzung: „Hi-Res" von High Resolution). In diesem Zustand kann der Apple 53 760 Punkte in einer 280 Punkte breiten und 192 Punkte hohen Matrix wiedergeben. Der Bildschirm kann schwarze, weiße, violette, grüne, rote und blaue Punkte anzeigen, obwohl es einige Einschränkungen gibt, was die Farbe der einzelnen Punkte betrifft.

Die hochauflösende Graphik nimmt ihre Daten aus einem 8 192 Byte großen Speicherbereich, der auch „Bildpuffer" genannt wird. Es gibt zwei unterschiedliche Bildpuffer: einen für die erste Seite und einen für die zweite Seite. Beide Puffer sind unabhängig und getrennt von den Speicherbereichen, die für die Wiedergabe von Text und Lo-Res-Graphik dienen. Der Bildpuffer für die erste Seite beginnt ab Adresse 8192 und erstreckt sich bis Adresse 16383 (hexadezimal: $4000 bis $5FFF). Ist Ihr Apple mit 16K (16384 Bytes) oder weniger Speicherplatz ausgerüstet, so ist die zweite Seite unerreichbar für Sie; liegen weniger als 16K Speicher vor, so müssen Sie ganz auf die hochauflösende Graphik verzichten. Jeder Punkt auf dem Bildschirm repräsentiert ein Bit aus dem Bildpuffer. Aus jedem Byte werden 7 Bit zur Anzeige gebracht, wobei das übrige Bit über die Farben der Punkte des Bytes bestimmt. Vierzig Bytes erscheinen auf jeder Zeile des Bildschirms. Das niedrigste Bit (Bit 0) des ersten Bytes einer Zeile erscheint ganz links auf dem Bildschirm. Darauf folgen Bit 1, Bit 2 u.s.w. Das höchste Bit (Bit 7) wird nicht angezeigt. Dann kommt das niedrigste Bit des nächsten Bytes u.s.w. So werden in jeder Zeile insgesamt 280 Punkte angezeigt.

Auf einem Schwarz-Weiß-Monitor erscheinen die Punkte weiß, deren Bit „an" (also „1") ist und die Punkte schwarz, deren Bit „aus" (also „0") ist. Auf einem Farbbildschirm sind die Verhältnisse nicht so einfach. Ist ein Bit aus, so gibt es wieder einen schwarzen Punkt, ist es an, so kommt es bei der Farbe auf die Position des Punktes auf dem bildschirm an. Ist der Punkt ganz links auf dem Bildschirm auf Spalte 0 oder auf einer anderen Spalte mit gerader Nummer, so wird er violett erscheinen. Befindet sich der Punkt in der am weitesten rechts liegenden Spalte, also Spalte 279 oder in einer Spalte mit ungerader Nummer, so erscheint der Punkt grün.

Zwei Punkte, die Seite an Seite stehen, erscheinen beide weiß. Ist das nicht angezeigte Bit eines Bytes „an", so ersetzen blau und rot die Farben violett und grün*. Es gibt also sechs verfügbare Farben in der hochauflösenden Graphik, die den folgenden Einschränkungen unterworfen sind:

1. Punkte in geraden Spalten sind schwarz, weiß, violett oder blau.

2. Punkte in ungeraden Spalten sind schwarz, weiß, grün oder rot.

3. Jedes Byte ist entweder ein Violett/Grün-Byte oder ein Blau/Rot-Byte. Es ist nicht möglich, Grün und Blau, Grün und Rot, Violett und Blau oder Violett und Rot im selben Byte zu mischen.

4. Zwei farbige Punkte „Seite an Seite" erscheinen immer weiß, sogar wenn sie unterschiedlichen Bytes gehören.

5. Auch bei europäischen Modellen des Apple gelten diese Regeln, aber die Farben der hochauflösenden Graphik können geringfügig abweichen.

Schaubild 3 zeigt den Bildschirm des Apple in hochauflösender Graphik mit den Speicheradressen jeder Zeile des Bildschirms.

* Auf Apple-Platinen der Version 0 sind die Farben rot und blau nicht verfügbar und das Setzen des höchsten Bits ist ohne Bedeutung.

Andere Ein-/Ausgabeeinrichtungen

Apple Ein-/Ausgabeeinrichtungen

Eingänge:	Cassetten-Eingang
	3 Ein-Bit-Digital-Eingänge
	4 Analog-Eingänge
Ausgänge:	Cassetten-Ausgang
	Eingebauter Lautsprecher
	4 "Signal"-Ausgänge
	Impulsausgang

Der Lautsprecher

Im Gehäuse des Apple befindet sich links unter der Tastatur ein kleiner 8 Ohm-Lautsprecher. Er ist so an die interne Elektronik des Apple angeschlossen, daß Programme ihn veranlassen können, verschiedene Geräusche abzugeben.

Der Lautsprecher wird von einem Programmschalter kontrolliert. Er kann die Membrane des Lautsprechers in zwei Positionen bringen: „innen" und „außen". Dieser Programmschalter arbeitet nicht wie die Programmschalter für die unterschiedlichen Bildausgaben, sondern funktioniert als Wechselschalter. Jedesmal, wenn ein Programm auf die Speicheradresse, die dem Lautsprecher zugeordnet ist, zugreift, wechselt die Membrane des Lautsprechers ihre Stellung von „innen" nach „außen" oder umgekehrt. Bei jedem Positionswechsel produziert der Lautsprecher ein kurzes „Klick". Ein Programm kann durch schnellen periodischen Zugriff auf die Adresse des Lautsprechers einen Dauerton erzeugen.

Der Programmschalter für den Lautsprecher ist der Speicherzelle 49200 zugeordnet. Jeder Zugriff auf diese Adresse (die gleichwertigen Adressen sind –16336 oder hexadezimal $C030) läßt den Lautsprecher ein „Klick" aussenden.

Mit einem Lese- oder Schreibvorgang kann ein Programm auf die Adresse der speziellen Speicherstelle für den Lautsprecher zugreifen. Die Daten, die gelesen oder geschrieben werden, sind ohne Bedeutung, denn es ist die Adresse, die den Schalter betätigt. Dabei ist zu berücksichtigen, daß einem Schreibvorgang im 6502-Mikroprozessor erst ein Lesevorgang vorangeht. So löst ein Schreibvorgang in die Speicherstelle eines Programmschalters das zweimalige Setzen dieses Schalters aus. Bei Wechselschaltern ergibt sich deshalb nach dem Schreibvorgang der alte Zustand vor dem Zugriff.

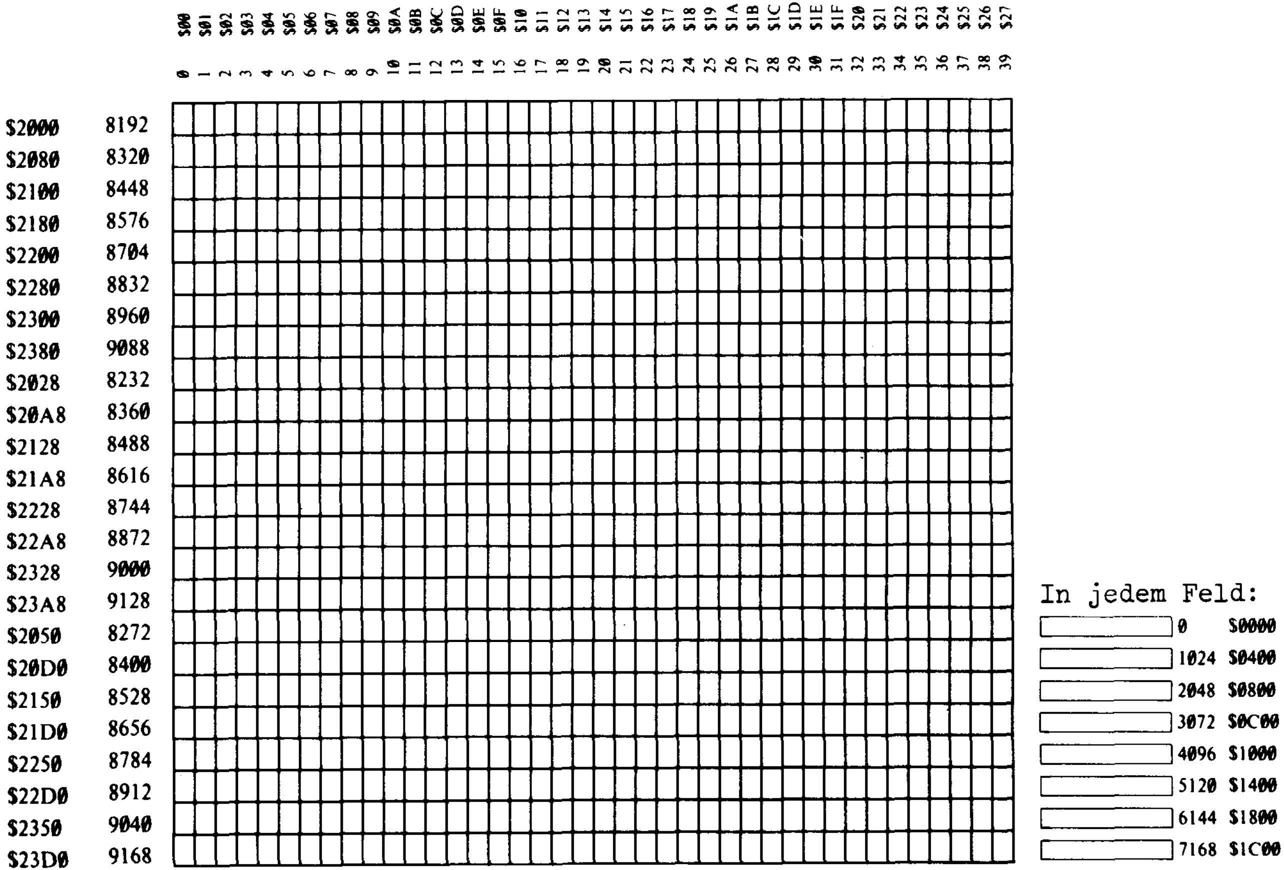


Schaubild 3: Speicherplan für die hochauflösende Graphik

Sie erhalten die Adresse für jedes Byte durch Addition der Reihe, der Spalte und der Zeile in dem Feld, in dem dieses Byte liegt.

Die Kassetten-Schnittstelle (Interface)

Rechts auf der hinteren Kante der Apple-Hauptplatine befinden sich zwei kleine schwarze mit „IN" und „OUT" bezeichnete Buchsen. Dies sind Miniatur-Phono-Buchsen, in die Sie ein Kabel stecken können, das an jedem Ende zwei Miniatur-Phono-Stecker hat. Das andere Ende des Kabels kann mit einem Standard-Kassetten-Rekorder verbunden werden, so daß Ihr Apple Informationen auf Musik-Kassetten schreiben und wieder lesen kann.

Der mit „OUT" bezeichnete Anschluß ist mit einem weiteren Programmschalter verbunden. Es handelt sich wie bei dem Lautsprecher um einen Wechselschalter (siehe oben).

Der Programmschalter für den Kassettenausgang kann durch Zugriff auf die Speicherzelle 49184 (entsprechend −16352 oder hexadezimal $C020) betätigt werden. Der Zugriff auf diese Adresse stellt die Spannung am Kassettenausgang von Null auf 25 mV oder läßt sie von 25 mV auf Null fallen. Steckt das andere Ende des Verbindungskabels im Mikrophoneingang des Kassettenrekorders und nimmt der Rekorder auf, so wird ein leises „Klick" aufgenommen. Ein Programm kann einen Ton durch periodischen Zugriff auf die Speicherstelle 49184 in der Aufnahme erzeugen. Durch Variation von Dauer und Tonhöhe kann Information für späteren Gebrauch verschlüsselt auf Band geschrieben werden. Der System-Monitor enthält solch ein Programm zur Kodierung von Daten für die Aufnahme auf Band (siehe Seite 50).

Es wird davor gewarnt, den Programm-Schalter durch Schreib-Vorgänge zu bedienen. Aus den in der Beschreibung des Lautsprechers erwähnten Gründen erzeugen Schreib-Vorgänge tatsächlich zwei „Klicks" auf der Aufnahme. Sie sollten nur Lese-Vorgänge zum Betätigen des Programmschalters verwenden.

Der andere, mit „IN" bezeichnete Anschluß, kann zum Abhören von Kassettenaufnahmen gebraucht werden. Sein Hauptzweck liegt darin, eine Einrichtung zu versorgen, die Töne vom Band hört, sie zu Daten dekodiert und diese im Speicher ablegt. So können auf Kassette gespeicherte Programme oder Daten gelesen und dann wieder verwendet werden.

Der Eingangs-Schaltkreis nimmt ein 1 V-Signal (Spitzenspannung) aus der Ohrhörer-Buchse und formt es in eine Folge von Einsen und Nullen um. Jedesmal, wenn das Signal am Eingang von Plus nach Minus wechselt oder umgekehrt, verändert der Eingangsschaltkreis seinen Zustand: von „1" nach "0" bzw. von „0" nach „1". Ein Programm kann durch Prüfen des Inhalts der Speicherstelle 49248 (entsprechend −16288 oder hexadezimal $C060) den Zustand des Eingangsschaltkreises feststellen. Ist der Inhalt dieser Speicherstelle kleiner als 128, so ist der Zustand Null, sonst liegt der Zustand Eins vor. Obwohl BASIC-Programme den Zustand des Kassetten-Eingangskreises lesen können, ist die Geschwindigkeit von BASIC-Programmen viel zu klein, um aus den einzeln „aufgepickten" Daten klug zu werden. Es gibt jedoch ein Programm im System-Monitor, das Töne von Kassetten liest und sie dekodiert (siehe Seite 51).

Ein-/Ausgabeanschluss für Spiele

Der Sinn dieses Anschlusses liegt darin, den Effekt von Programmen, insbesondere von Spielprogrammen, zu erhöhen. Dazu stehen Ihnen eine Reihe von Ein- und Ausgängen zur Verfügung, an die Sie spezielle Geräte anschließen können. Es handelt sich um drei Ein-Bit-Eingänge, vier Ein-Bit-Ausgänge, einen Impuls-Ausgang und vier analoge Eingänge, die alle von Ihren Programmen gesteuert werden können. Ihr Apple ist mit zwei Spielsteuerungen (Game Controller) ausgestattet, deren Anschlußkabel im Spiele-Anschluß stecken. Die zwei Drehknöpfe der Steuerungen sind mit zwei Analogeingängen verbunden. Die zwei Drucktasten sind an zwei Ein-Bit-Eingänge des Spiele-Anschlusses angeschlossen.

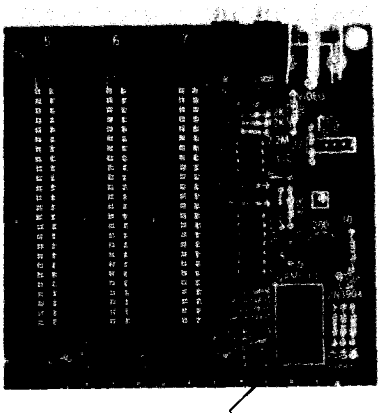


Photo 6. Ein-/Ausgabeanschluß für Spiele.

Signal-Ausgänge

Die Signal-Ausgänge sind die vier Ein-Bit-Ausgänge. Jeder Signal-Ausgang kann anderen elektronischen Geräten als Eingang dienen oder an Schaltkreisen angeschlossen werden, die Lampen, Relais, Lautsprecher oder andere Geräte treiben.

Jeder Signal-Ausgang wird von einem Programm-Schalter gesteuert, der auf zwei Adressen ansprechbar ist. So gibt es insgesamt 8 Adressen, wobei der Zugriff auf die eine Adresse den Ausgang auf „an" stellt und ein Zugriff auf die andere Adresse den Ausgang auf „aus" stellt. Ein Signal-Ausgang hat an seinem Anschluß eine Spannung von fast 0 V, wenn er „aus" ist, sonst liegt eine Spannung von fast +5 V an. Es ist keine Möglichkeit vorgesehen, den aktuellen Zustand eines Signal-Ausgangs festzustellen. Dies sind die Programmschalter:

Tabelle 9: Adressen für die Signal-Ausgänge				
Nr.	Zustand	Adresse		
		Dezimal		Hexa
∅	aus	49240	-16296	$C∅58
	an	49241	-16295	$C∅59
1	aus	49242	-16294	$C∅5A
	an	49243	-16293	$C∅5B
2	aus	49244	-16292	$C∅5C
	an	49245	-16291	$C∅5D
3	aus	49246	-16290	$C∅5E
	an	49247	-16289	$C∅5F

Ein-Bit-Eingänge

Die drei Ein-Bit-Eingänge können entweder an andere elektronische Geräte oder an Druckstasten angeschlossen werden. Sie können den Zustand jedes Eingangs mit Programmen in Maschinensprache oder BASIC auf dieselbe Weise erfahre, wie Sie den Kassetteneingang lesen. Die Speicherstellen für die drei Ein-Bit-Eingänge haben die Adressen 49249 bis 49251 (–16287 bis –16285 oder hexadezimal $C∅61 bis $C∅63).

Analog-Eingänge

Die vier Analog-Eingänge können an 150 kOhm-Potentiometer angeschlossen werden. Der variable Widerstand zwischen jedem Eingang und +5 V steuert einen Ein-Chip-Zeitkreis. Bei Änderung des Widerstandes an einem Eingang verändert sich entsprechend die Zeitcharakteristik des zugehörigen Zeitkreises. Programme in Maschinensprache können Veränderungen in den Zeitschleifen wahrnehmen und verschaffen sich so einen numerischen Wert, der der Stellung des Potentiometers entspricht.

Bevor ein Programm beginnen kann, die Position der Potentiometer zu ermitteln, muß es erst die Zeitkreise zurückstellen, indem es auf den Programmschalter 49264 (–16272 oder hexadezimal $C∅7∅) zugreift. Beim Zurücksetzen der Zeitkreise werden die Werte in den vier Speicherstellen 49252 bis 49255 (–16284 bis –16281 oder $C∅64 bis $C∅67) größer als 128 (Bit 7 wird gesetzt). Innerhalb von 3,∅6∅ Millisekunden sollten die Werte dieser vier Speicherstellen unter 128 fallen. Die genaue Zeit, die jede Speicherstelle braucht, um unter den Wert 128 zu fallen, ist direkt proportional zu der Stellung des Potentiometers, das zu dieser Speicherstelle gehört. Falls das angeschlossene Potentiometer einen größeren Widerstand als 15∅ kOhm hat oder kein Potentiometer angeschlossen ist, fällt der Wert in der zugeordneten Speicherstelle nicht auf Null.

Impuls-Ausgang

Es gibt noch einen zusätzlichen Ausgang, den Impuls-Ausgang, der normalerweise auf +5 V liegt, aber dessen Spannung durch den Zugriff auf die Adresse 49216 (–16320 oder $C04F) für die Dauer einer halben Mikrosekunde auf 0 V fällt. Man beachte, daß ein Schreib-Vorgang hier zwei Impulse auslöst. (Der Abschnitt über den Lautsprecher beschreibt dieses Phänomen.)

Tabelle 10: Ein-/Ausgabe-Adressen

Funktion	Adresse			Lesen/Schreiben
	Dezimal		Hexa	
Lautsprecher	49200	-16336	$C030	L
Kassetten-Eingang	49256	-16288	$C060	L
Kassetten-Ausgang	49184	-16352	$C020	L
Signal-Ausgänge	49240	-16296	$C058	L/S
	bis	bis	bis	
	49247	-16289	$C05F	
Ein-Bit-Eingänge	49249	-16287	$C061	L
	49250	-16286	$C062	L
	49251	-16285	$C063	L
Analog-Eingänge	49252	-16284	$C064	L
	49253	-16283	$C065	
	49254	-16282	$C066	
	49255	-16281	$C067	
Analog Rücksetzen	49264	-16272	$C070	L/S
Impuls-Ausgang	49216	-16320	$C040	L

Variationen des Apple

Es gibt ein paar Variationen des Apple II Computers. Manche der Variationen sind Verbesserungen oder Modifikationen des Computers an sich, andere sind Änderungen an Betriebsprogrammen. Das sind die grundlegenden Variationen.

Autostart-ROM / Monitor-ROM

Alle Apple II-Plus-Systeme enthalten den Autostart Monitor-ROM. Alle anderen Apple-Systeme besitzen keinen Autostart-ROM, haben dafür aber den Apple System-Monitor-ROM. Dieser ROM-Version fehlen einige Einrichtungen, die im Autostart-ROM enthalten sind, sie hat aber auch Einrichtungen, die es nicht im Autostart-ROM gibt. Die Hauptunterschiede zwischen den beiden ROM's werden auf den nächsten Seiten erläutert.

- **Textredigieren.** Die ESC-I, -J, -K und -M Tastenfolgen, die den Zeiger nach oben, links, rechts oder unten bewegen, sind im alten Monitor-ROM nicht verfügbar.

- **Ausgabestop.** Die Möglichkeit, die Ausgabe der meisten BASIC- oder Maschinensprachen-Programme oder Programmauflistungen mit CTRL S zu stoppen, besteht erst im neuen Monitor-ROM.

- **Der RESET-Ablauf.** Beim Apple mit altem Monitor-ROM meldet sich beim Einschalten oder Betätigen der RESET -Taste das Monitor-System anstatt einen „Warm-" oder „Kaltstart" einzuleiten, wie er auf Seite 39 („Der RESET -Ablauf") beschrieben ist.

Im alten Monitor-ROM gibt es noch die STEP- und TRACE-Fehlerhilfen des System-Monitors (siehe Seite 55). Das Autostart-ROM erkennt diese Monitor-Kommandos nicht.

Hauptplatine Version 0 / Version 1

Die Version 0 und die derzeitige Version 1 der Hauptplatine des Apple II unterscheiden sich in wenigen Punkten. Um festzustellen, um welche Art Platine es sich in Ihrem Apple handelt, heben Sie die Abdeckung und vergleichen Sie die rechte obere Ecke der Platine mit dem Photo 4 auf Seite 10. Sie besitzen die Version 0 des Apple, wenn Sie den einzelnen metallenen Video-Verbindungsstift zwischen der Vier-Stift Video-Anschluß-Buchse und dem runden Bildaussteuerungstrimmer nicht finden.

Die Unterschiede zwischen den beiden Versionen des Apple sind:

- **Farbunterdrückung.** Im Text-Modus läßt die Apple-Version 0 das Farbsignal am Video-Ausgang aktiv. Dadurch erscheinen die Zeichen auf dem Bildschirm getönt oder mit farbigen Rändern.

- **Einschalt-RESET.** Die Version 0 hat keine Schaltung, um automatisch RESET einzuleiten, wenn Sie das Gerät einschalten. Stattdessen müssen Sie selbst RESET drücken.

  Beim Einschalten des Apple der Version 0 enthält das Tastenfeld irgendein Zeichen. Da der Apple eine Eingabe erwartet, behandelt er dieses Zeichen, als wäre es von Ihnen eingegeben. Um dieses Zeichen zu löschen, empfiehlt es sich, beim Einschalten nach dem RESET ein CTRL X zu geben.

- **Farben in der hochauflösenden Graphik.** Der Apple der Version 0 hat nur vier Farben für die hochauflösende Graphik: schwarz, weiß, violett und grün. Das höchste Bit (Bit 7) eines jeden auf den Bildschirm gebrachten Bytes wird ignoriert (siehe Seite 20).

- **24K Speicheraufteilungs-Problem.** Systeme der Version 0, die 20 oder 24K Bytes RAM besitzen, scheinen über mehr Speicherplatz für BASIC zu verfügen, als sie tatsächlich haben (siehe Seite 76).

- **50 Hz-Apple.** Bei der Hauptplatine Version 0 läßt sich das Video-Ausgangssignal nicht auf das europäische PAL-/SECAM-Fernseh-System umstellen. Außerdem fehlt der dritte Video-Anschluß zwischen den oben erwähnten vier Stiften der Video-Verbindung und dem runden Bildaussteuerungstrimmer.

- **Lautsprecher bei Kassettenbetrieb.** Jeder Ton, der vom eingebauten Lautsprecher abgegeben wird, erscheint bei der Version 0 des Apple auch auf dem Ausgang des Kassettenanschlusses. Befindet sich der Rekorder im Zustand der Aufnahme (RECORD), so wird jeder Ton des Lautsprechers auch aufgezeichnet.

- **Kassetten-Wiedergabe (Einlesen).** Der Eingangsschaltkreis des Kassettenanschlusses wurde so verändert, daß schwächere Signale mit größerer Ganauigkeit erkannt werden.

Verbessertes Netzteil

Manche Apple-Geräte haben ein Netzteil, das nur mit 110 Volt betrieben werden kann, während andere Versionen mit einem Spannungswahlschalter (110/220 V) an der Rückseite des Gerätes ausgerüstet sind.

Der Apple II Plus

Der Apple II Plus ist ein Standard-Apple II-Computer mit der Hauptplatine Version 1, dem Autostart-ROM und dem Applesoft II BASIC (Fließkomma-BASIC) im ROM anstatt dem Apple Integer BASIC (Ganzzahl-BASIC). Europäische Modelle des Apple II Plus sind mit einem 110/220 V-Netzteil ausgestattet. Der Apple Mini-Assembler, das Fließkomma-Programmpaket und der Sweet 16-Interpreter, die sonst im Apple Integer BASIC ROM gespeichert sind, sind auf dem Apple II Plus nicht verfügbar.

KAPITEL 2
Der Dialog mit dem Apple

Fast alle Programme und Sprachen auf dem Apple brauchen Eingaben von der Tastatur und Möglichkeiten, Informationen auf den Bildschirm zu schreiben. Dazu stehen im ROM-Speicher des Apple einige Unterprogramme zur Verfügung, die die meisten Ein-/Ausgabeprozesse in allen Programmen und Sprachen im Apple durchführen.

Die Unterprogramme für diese Ein-/Ausgabefunktionen haben verschiedene Namen, die sich die Autoren dieser Programme ausgedacht haben. Der Apple selbst kennt die Namen seiner Maschinenunterprogramme nicht, trotzdem ist es für uns alle wesentlich bequemer, diese Unterprogramme bei ihren Namen zu nennen.

Standard-Ausgabe

Das Standard-Ausgabe-Unterprogramm heißt COUT (Character OUT). COUT schreibt Großbuchstaben in normal oder invers auf den Bildschirm. Es ignoriert die Steuerzeichen außer RETURN, dem Zeilenvorschub (line feed), Rückwärtsschritt und Klingelzeichen (bell).

Das COUT-Unterprogramm hat einen eigenen unsichtbaren „Ausgabe-Zeiger" (Die Position, auf die das nächste Zeichen gesetzt werden soll). Jedes Mal, wenn COUT aufgerufen wird, setzt es ein Zeichen auf die aktuelle Zeigerposition auf dem Bildschirm (und überschreibt damit das Zeichen, das dort vorher stand). Dann wird der Zeiger um eine Stelle nach rechts bewegt. Geht er dabei über die rechte kante des Bildschirms, wird er auf die erste Stelle der nächsten Zeile gesetzt. Verläßt der Zeiger die unterste Zeile, so wird der gesamte Bildschirminhalt um eine Zeile nach oben geschoben und der Zeiger auf die erste Stelle der nun leeren untersten Zeile gesetzt.

Wenn ein RETURN an COUT ausgegeben wird, setzt es den Zeiger an den Anfang der nächsten Zeile. Verläßt der Zeiger dabei die letzte Zeile, so wird das Bild wie oben beschrieben um eine Zeile hochgeschoben.

Ausgabe-Stop

Wenn ein Programm oder ein Sprache ein RETURN zu COUT schickt, sieht COUT zuerst in der Tastatur nach, ob Sie seit dem letzten Aufruf CTRL S gedrückt haben. Wenn das der Fall ist, hält das Programm an und wartet auf einen weiteren Tastendruck von Ihnen. Diese Einrichtung wird „Ausgabe-Stop" genannt*. Betätigen Sie dann eine Taste, so gibt COUT das RETURN -Zeichen aus und setzt seine normale Ausgabe fort. Der Code der gedrückten Taste wird ignoriert, es sei denn, es handelt sich um CTRL C. In diesem Fall gibt COUT dem Programm oder der Sprache diesen Zeichencode weiter. So ist es Ihnen durch ein CTRL C im Ausgabestop möglich, Programmauflistungen oder BASIC-Programme zu beenden.

Ein Zeilenvorschub-Zeichen bewirkt, daß COUT seinen Zeiger ohne waagrechte Verschiebung um eine Zeile tiefer stellt. Wie immer wird das Bild nach oben geschoben, wenn der Zeiger unter die letzte Zeile kommt. Der Zeiger bleibt dabei auf seiner Position in einer neuen untersten Zeile.

* Die Ausgabestop-Einrichtung ist auf Apples ohne Autostart-ROM nicht vorhanden.

Ein Rücksetz-Zeichen (durch ← ausgelöst) bewegt den unsichtbaren Zeiger um eine Stelle nach links. Geht der Zeiger über die linke Kante, so wird er ganz rechts in die vorhergehende Zeile gesetzt. Gibt es keine solche Zeile (weil der Zeiger oben auf dem Bildschirm ist), so bleibt er ganz rechts in der obersten Zeile, ohne daß der Bildinhalt verschoben wird.

Soll COUT ein Klingel-Zeichen (bell) ausgeben, so wird der Bildschirm nicht verändert und der Zeider nicht verschoben. Stattdessen wird für 1/10 Sekunde ein Ton von 1kHz vom Lautsprecher ausgegeben.

Das Text-Fenster

In der obigen Diskussion über die verschiedenen Bewegungen des Ausgabe-Zeigers bezogen sich die Angaben „links", „rechts", „Oben" Und „unten" auf das linke, rechte, obere und untere Ende eines normalen 40 Zeichen breiten und 24 Zeichen hohen Bildschirms. Es gibt jedoch die Möglichkeit, dem Unterprogramm COUT mitzuteilen, daß es nur in einem bestimmten Bildschirmbereich arbeiten soll und nicht auf der ganzen 960-Zeichen-Anzeige.

Dieser abgesonderte Text-Bereich heißt „Fenster". Programme oder Sprachen können die vier Angaben „Oben", „Unten", „linke Seite" und „Breite" des Textfensters in vier spezielle Speicherstellen geben. Dann wird COUT die Größe des Bildschirms aus diesen neuen Angaben berechnen und nie Text außerhalb des Fensters drucken. Soll der Bildschirm hochgeschoben werden, so geschieht dies nur innerhalb des Textfensters. Auf diesem Weg haben Programme Einfluß auf die Unterbringung von Text auf dem Bildschirm und können damit andere Bereiche davor schützen, von neuem Text überschrieben zu werden.

Die Speicherstelle 32 (hexadezimal $20) enthält die Spaltenposition ganz links im Fenster. Es ist normalerweise Spalte 0 ganz links auf dem Bildschirm. Diese Zahl sollte nie 39 (hexadezimal $27) überschreiten, das wäre schon die äußerste Spalte rechts. Die Speicherstelle 33 (hexadezimal $21) bestimmt die Breite des Textfensters (Anzahl der Spalten). Beachten Sie, daß die Summe aus der linken Position und der Breite des Fensters nie 40 überschreitet! Das hätte zur Folge, daß COUT Zeichen in Speicherstellen außerhalb des Bildschirmbereichs schreibt und damit Ihre Programme und Daten gefährdet.

Die Speicherstelle 34 (hexadezimal $22) enthält die Nummer der obersten Zeile des Textfensters. Normalerweise ist es 0, also die oberste Zeile des Bildschirms. Schließlich gibt die Speicherstelle 35 (hexadezimal $23) die (um 1 erhöhte) maximale Zeilennummer an. Normalerweise ist es 24 für die unterste Zeile des Bildschirms. (Die Rechnung geht auf: 23 + 1 = 24. Die Zeile 23 ist aber die 24. Zeile, da immer* von 0 bis 23 gezählt wird.)

Wenn Sie das Textfenster verändern, sollten Sie darauf achten, daß Sie wissen, wo der Ausgabe-Zeiger gerade ist, damit er sich innerhalb des neuen Fensters befindet.

---

* Jede Speicherstelle hat wenigstens ein Bit (sonst ist es keine Speicherstelle). Damit können mindestens die Werte 0 und 1 gespeichert werden. Das sind schon 2 Zustände, wobei der 1. Zustand der „Zustand 0" ist und der zweite der „Zustand 1". So kommt es, daß ein Computer von „Natur aus" beim Zählen immer mit 0 anfängt. Also nochmal: die erste Zeile ist „Zeile 0", die letzte Zeile ist „Zeile 23".

Tabelle 11: Speicherstellen für das Text-Fenster

Funktion	Adresse		Minimum/Normal/Maximum	
	Dezimal	Hexa	Dezimal	Hexa
Linke Kante	32	$2Ø	0/0/39	$Ø/$Ø/$17
Breite	33	$21	0/40/40	$Ø/$28/$28
Obere Kante	34	$22	0/0/24	$Ø/$Ø/$18
Untere Kante	35	$23	0/24/24	$Ø/$18/$18

Zeichendarstellung

Das COUT-Unterprogramm kann Zeichen „Normal" oder „Invers" ausdrucken. Die spezielle Ausgabeart wird vom Inhalt der Speicherstelle 5Ø (hexadezimal $32) bestimmt. Steht dort der Wert 255 (hexadezimal $FF), so werden die Zeichen von COUT im Normal-Modus ausgegeben. Inverse Zeichen entstehen, wenn 63 (hexadezimal $3F) der Inhalt dieser Speicherstelle ist. Beachten Sie, daß ein Wechsel des Ausgabezustandes nur auf die Zeichen Einfluß hat, die nach dem Wechsel gedruckt werden. Weitere Werte in der Speicherstelle 5Ø verursachen ungewöhnliche Dinge: beim Wert 127 erscheinen blinkende Buchstaben und alle anderen Zeichen invers. Bei allen anderen Werten ignoriert COUT einige oder alle Zeichen des normalen Zeichensatzes.

Tabelle 12: Ausgabekontrolle durch Speicherstelle 50

Wert		Effekt
Dezimal	Hexa	
255	$FF	COUT gibt Zeichen im "NORMAL"-Modus aus.
63	$3F	COUT gibt Zeichen im "INVERSE"-Modus aus.
127	$7F	COUT gibt Buchstaben im "FLASH"-Modus aus und alle anderen Zeichen im "INVERSE"-Modus.

Die Speicherstelle 5Ø enthält eine sog. „Maske", die durch ein logisches „UND" (AND) die einzelnen Bits jedes auszugebenden Zeichencodes verändert. Jedes Bit dieser „Maske" mit dem Wert „Ø" läßt das Bit an derselben Stelle im Zeichencode auch eine „Ø" werden, unabhängig von dem Wert, den es zuvor hatte. So werden die beiden oberen Bits (Bit 7 und Bit 6) eines jeden Ausgabezeichens auf „Ø" gesetzt, wenn die Speicherstelle 5Ø eine 63 (hexadezimal $3F oder binär ØØ111111) enthält. Auf dem Bildschirm erscheinen dann Zeichen, deren Codes zwischen Ø und 63 liegen. Wie man der Bildschirmcode-Tabelle auf Seite 15 entnimmt, handelt es sich also um Zeichen im „INVERSE"-Modus.

Standard-Eingabe

Hier gibt es sogar zwei Unterprogramme, die sich mit der Sammlung von Standard-Eingaben befassen: RDKEY (Read a key) holt einen einzigen Tastendruck aus der Tastatur und GETLN (Get a line) sammelt eine Reihe von Tastendrucken zu einer „Eingabezeile" zusammen.

RDKEY

In erster Linie wartet das RDKEY-Unterprogramm darauf, daß der Benutzer eine Taste der Tastatur betätigt, um dem Programm, das es aufgerufen hat, den Code der gedrückten Taste weiterzugeben. Während dieser Tätigkeit verrichte RDKEY noch zwei andere nützliche Dienste:

1. **Hinweis auf die Eingabestelle.**
   Beim Aufruf von RDKEY wird zuerst der unsichtbare Ausgabe-Zeiger sichtbar gemacht. Dies weist den Benutzer darauf hin, daß der Apple auf einen Tastendruck wartet und es bringt die erfragte Eingabe mit einer bestimmten Bildschirmposition in Beziehung. Meist erscheint der Eingabehinweis in der Nähe eines Textes, der beschreibt, was das spezielle Programm (oder die Sprache) als Eingabe erwartet. Der Eingabe-Zeiger selbst ist eine blinkende Wiedergabe des Zeichens, das sich auf der Stelle des Ausgabezeigers befindet. Gewöhnlich ist es das Leerzeichen, wodurch der Eingabezeiger als blinkendes Quadrat erscheint.

   Wenn der Benutzer eine Taste drückt, entfernt RDKEY den Eingabezeiger und kehrt mit dem Wert der gedrückten Taste zu dem Programm zurück, das es aufgerufen hat. Bedenken Sie, daß der Ausgabezeiger nur eine Position, während der Eingabezeiger ein blinkendes Zeichen auf dem Bildschirm darstellt. Beide bewegen sich immer auf dieselbe Weise und sind kaum voneinander getrennt, aber wenn der Eingabezeiger verschwindet, bleibt der Ausgabezeiger noch aktiv.

2. **Erzeugen von Zufallszahlen.**
   Während RDKEY auf einen Tastendruck des Benutzers wartet, addiert es fortlaufend eine 1 zu einem Speicherstellenpaar. Wird schließlich eine Taste betätigt, so stellen diese beiden Speicherstellen zusammen eine Zahl zwischen 0 und 65535 dar. Der genaue Wert ist völlig unvorhersehbar. So können viele Sprachen und Programme ihn als Basis für einen Zufallszahlengenerator verwenden.

   Die Speicherstellen der Zufallszahl, die RDKEY laufend verändert, finden sich auf den Adressen 78 und 79 (hexadezimal $4E und $4F).

GETLN

Die meisten Eingaben auf dem Apple werden zu Eingabezeilen zusammengefaßt. Das Unterprogramm GETLN im ROM des Apple erfragt eine Eingabezeile von der Tastatur und kehrt, nachdem es eine bekommen hat, zu dem Programm zurück, das GETLN aufgerufen hat. GETLN hat viele Einrichtungen und Nuancen, und es ist gut, alle Funktionen von GETLN zu kennen.

Beim Aufruf druckt GETLN erst das Bereitschaftszeichen. Es hilft Ihnen festzustellen, welches Programm durch GETLN eine Eingabe erfragt. Ein Stern (*) als Bereitschaftszeichen steht für den System-Monitor, eine rechte Spitze (>) für das Apple Integer BASIC, eine rechte eckige Klammer (]) für Applesoft II BASIC und das Ausrufezeichen (!) für den Apple Mini-Assembler. Zusätzlich wird das Fragezeichen (?) von vielen Programmen und Sprachen benutzt, um anzuzeigen, daß ein Benutzer-Programm eine Eingabe erwartet. Was Sie als Benutzer sehen, ist, daß der Apple einfach ein Bereitschaftszeichen druckt und einen Eingabezeiger zur Anzeige bringt. Schreiben Sie, so erscheinen die Zeichen der gedrückten Tasten auf dem Bildschirm und der Zeiger bewegt sich mit. Drücken Sie die RETURN -Taste, so wird die gesamte Zeile zu dem Programm oder zu der Sprache, zu der Sie sprechen, geschickt und Sie erhalten eine weitere Aufforderung.

Diese Vorgänge laufen aber in Wirklichkeit wesentlich komplizierter ab. GETLN druckt das Bereitschaftszeichen und ruft RDKEY auf, das den Eingabezeiger anzeigt. Wenn RDKEY mit einem Tastencode zurückkehrt, speichert ihn GETLN in einen Eingabepuffer und schreibt das Zeichen dort auf den Bildschirm, wo der Eingabezeiger stand. Dann ruft es wieder RDKEY auf. Dieser Vorgang wiederholt sich, bis der Benutzer die RETURN -Taste drückt. Wenn GETLN dieses RETURN vom Tastenfeld erhält, setzt es das Zeichen an das Ende des Eingabepuffers, löscht den Rest der Bildschirmzeile, in der der Eingabezeiger war und sendet ein RETURN zu COUT (siehe Seite 32). GETLN kehrt dann zu dem Programm zurück, das es aufgerufen hat. Programme oder Sprachen, die eine Eingabe erwartet haben, können sich nun die ganze Eingabezeile ansehen, da sie im Eingabepuffer steht.

Wann immer Sie eine Zeile schreiben, können Sie ein CTRL X geben und damit die gesamte Zeile löschen. GETLN wird einfach alles vergessen, was Sie getippt haben, ein „Backslash" (\) drucken, zu einer neuen Zeile springen, nochmal ein Bereitschaftszeichen geben und Ihnen so die Eingabe einer neuen Zeile ermöglichen. Eine Eingabezeile darf maximal 255 Zeichen lang sein. Wenn Sie diese Grenze überschreiten, wird GETLN die gesamte Zeile löschen und Sie müssen nocheinmal von vorne anfangen. Damit Sie vor dieser Grenze gewarnt sind, gibt GETLN ab dem 249. Zeichen mit jedem Tastendruck einen Ton aus.

GETLN erlaubt Ihnen auch, Zeilen, die Sie schreiben, zu verändern und zu korrigieren, damit Sie einfache Tippfehler verbessern können (Eine Einführung in die Standard-Korrektur-Funktionen und den Gebrauch der beiden Pfeiltasten (← und →) erscheint auf den Seiten 28-29 und 53-55 des Apple II BASIC Programmierhandbuchs oder auf den Seiten 27-28 und 52-53 des Anhangs C in der Applesoft Programmiereinführung). Hier ist eine kurze Beschreibung der Korrektureinrichtungen von GETLN :

Die Linkspfeil- oder Rücksetz-Taste (←)

Jeder Druck auf diese Taste läßt GETLN ein dahinterliegendes Zeichen in der Eingabezeile vergessen. Es wird auch ein Rücksetzzeichen zu COUT geschickt (siehe oben), um die Zeiger auf das Zeichen zu stellen, das gelöscht wurde. Wenn Sie nun ein Zeichen tippen, werden das gelöschte Zeichen auf dem Bildschirm und das gelöschte Zeichen in der Eingabezeile durch das getippte Zeichen überschrieben. Mehrfache Betätigung der Linkspfeiltaste löscht aufeinanderfolgende Zeichen. Jedoch wird GETLN die ganze Zeile ignorieren und eine nochmalige Eingabe verlangen, falls Sie mehr Zeichen mit der Linkspfeiltaste löschen, als Sie zuvor eingegeben hatten.

Die Rechtspfeil- oder Wiedereingabe-Taste (→)

Die Betätigung der Rechtspfeiltaste hat denselben Effekt wie das Tippen des Zeichens, auf dem der Zeiger steht. Zur Korrektur von Tippfehlern ist dies besonders nützlich, um den Rest einer Zeile neu einzugeben, über die mit der Linkspfeiltaste zuvor zurückgesetzt wurde. In Verbindung mit den reinen Zeigerbewegungen (siehe nächster Abschnitt) kann man Daten nochmal eingeben oder korrigieren, die schon auf dem Bildschirm stehen.

Der Escape-Zustand

Wenn Sie die ESC -Taste drücken, gehen die Eingabe-Unterprogramme in den „ESCAPE-ZUSTAND" (Escape : einer Eingabe „entkommen"). In diesem Zustand haben elf Tasten neue Bedeutungen, die „Escape-Anweisungen". Wenn Sie eine dieser elf Tasten drücken, wird der Apple die Funktion ausführen, die mit der Taste im Escape-Zustand verbunden ist. Je nachdem, welche Escape-Anweisung ausgeführt wurde, verläßt der Apple den Escape-Zustand oder behält ihn bei. Falls Sie im Escape-Zustand eine Taste betätigen, die keine Escape-Anweisung ist, wird dieser Tastendruck ignoriert und der Escape-Zustand beendet.

Der Apple kennt elf Escape-Anweisungen. Acht davon sind reine Zeigerbewegungen, die nur den Zeiger bewegen, ohne das Bild oder die Eingabezeile zu ändern, und drei davon sind Löschbefehle, die nur Teile oder das ganze Bild löschen. Alle Löschbefehle und die ersten vier reinen Zeigerbewegungen (Escape-Anweisungen @, A, B, C, D, E und F) beenden nach der Ausführung den Escape-Zustand. Die restlichen vier Escape-Anweisungen lassen den Escape-Zustand aktiv*.

ESC A Ein ESC -Tastendruck, gefolgt von einer A -Taste, bewegt den Zeiger eine Stelle nach rechts, ohne die Eingabezeile zu ändern. So kann man unerwünschte Zeichen einer Zeile überspringen : gehen Sie mit der Linkspfeiltaste über die Zeichen hinweg, geben Sie ESC A für jedes falsche Zeichen und geben Sie schließlich mit dem Rechtspfeil den Rest der Zeile wieder ein.

ESC B Geben Sie ESC und dann ein B , so wird der Zeiger um ein Zeichen nach links gesetzt, ohne die eingabezeile zu verändern. Damit kann man etwas zweimal in einer Zeile eingeben, ohne es nochmal tippen zu müssen : Tippen sie es einmal und gehen Sie mit wiederholtem ESC B an den Anfang der Eingabe. Nun können Sie allein mit der Rechtspfeiltaste die Eingabe wiederholen.

ESC C Die ESC C Tastenfolge setzt den Zeiger ohne seitliche Bewegung eine Zeile tiefer. Erreicht der Zeiger das untere Ende des Textfensters, so wird wie üblich der Text im Fenster eine Zeile nach oben geschoben. Die Eingabezeile wird nicht von ESC C verändert. ESC C und ESC D (siehe unten) sind nützlich, um den Zeiger an den Anfang einer anderen Zeile auf dem Bildschirm zu setzen und diese mit der Rechtspfeiltaste nochmals einzugeben.

* Die vier Escape-Anweisungen I, J, K und M sind nicht in Apples ohne Autostart-ROM verfügbar.

ESC D Diese Tastenfolge bewegt den Zeiger ohne seitliche Verschiebung um eine Zeile nach oben. Der Zeiger bleibt stehen, wenn er das obere Ende des Textfensters erreicht. Die Eingabezeile bleibt unverändert. Mit **ESC D** ist es möglich, eine vorangehende Zeile mit dem Zeiger zu erreichen und sie mit der Rechtspfeiltaste erneut einzugeben.

ESC E Die **ESC E** Tastenfolge wird „Löschen bis Zeilenende" genaunt. Erkennt COUT diese Tastenfolge, so wird der Rest der Bildschirmzeile ab der Zeigerposition bis zum rechten Ende des Textfensters gelöscht. Der Zeiger bleibt an seiner alten Stelle und die Eingabe bleibt unverändert. **ESC E** löscht den Rest der Bildschirmzeile, unabhängig vom Normal/Invers-Modus, zu Leerzeichen.

ESC F „Lösche bis Bildende". Hier wird alles im Textfenster gelöscht, das rechts oder unterhalb des Zeigers steht. Die Eingabe wird nicht verändert und der Zeiger bleibt an seiner alten Stelle. Mit dieser Tastenfolge können Sie sich wieder freie Sicht verschaffen, nachdem der Bildschirm durch viele Zeigerbewegungen und Korrekturen völlig unübersichtlich geworden ist.

ESC @ Die **ESC @** Tastenfolge heißt „Bildschirm löschen" (genauer: „home and clear"). Das gesamte Textfenster wird gelöscht und der Zeiger erscheint dann links oben in der „Home"-Position. Auch hier wird unabhängig vom Normal/Invers-Zustand zu Leerzeichen gelöscht und die Eingabezeile nicht verändert (Bemerkung: „@" ist **SHIFT P**).

ESC K
ESC J
ESC M
ESC I

Diese vier Escape-Anweisungen entsprechen den vier reinen Zeigerbewegungen **ESC A**, **ESC B**, **ESC C** und **ESC D**, nur beenden sie nach der Ausführung den Escape-Zustand nicht. So können Sie fortgesetzt **A**, **B**, **C** bzw. **D** eintippen, ohne immer erst **ESC** eingeben zu müssen.

Sie beenden diesen Zustand erst mit einer anderen Escape-Taste (A, B, C, D, E, F oder @) oder durch irgendeine anderen Taste, außer der letzten Escape-Anweisung. Diese Taste wird dann als Eingabe ignoriert. Die vier Tasten K, J, M und I sind in einem Richtungsfeld auf der Tastatur angeordnet, so daß Sie aus der Lage der vier Tasten zueinander auf die Bewegung des Zeigers schließen können.

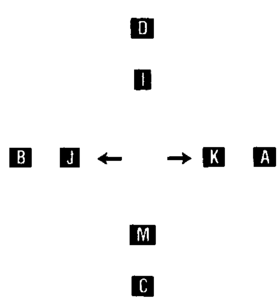


Schaubild 4. Escape-Anweisungen für Zeigerbewegungen.

Der Reset-Ablauf

Wenn Sie an Ihrem Apple den Netzschalter anschalten* oder die RESET -Taste drücken und wieder loslassen, startet der 65Ø2-Mikroprozessor des Apple einen RESET -Ablauf durch einen Sprung in ein Unterprogramm des Apple Monitor-ROM's. In den zwei unterschiedlichen Versionen dieses ROM's, dem Monitor-ROM und dem Autostart-ROM, werden im RESET -Ablauf sehr unterschiedliche Tätigkeiten ausgeführt.

Der Autostart-ROM Reset

Apples mit dem Autostart-ROM beginnen ihren RESET -Ablauf mit dem Einstellen der Bildschirmanzeige. Die Programmschalter werden auf die Ausgabe der ersten Seite im Text-Modus gestellt.

Unsichtbar hinter dem Text wird schon Block-Graphik mit gemischter Anzeige (vier Zeilen Text unter der Graphik) vorbereitet. Das Textfenster wird auf volle Größe geöffnet, der Ausgabe-Zeiger auf die unterste Zeile gesetzt und die Zeichenausgabe auf „Normal"-Modus (weiße Zeichen auf schwarzem Grund) gestellt. Dann werden die COUT- und KEYIN-Programmschalter auf den Gebrauch des Bildschirms und der Tastatur zur Standard-Ein-/Ausgabe gestellt, die Signal-Ausgänge Ø und 1 des Spieleanschlusses auf „An" und die Signal-Ausgänge 2 und 3 auf „Aus" geschaltet. Schließlich wird das Tastensignal gelöscht, die aktiven Ein/Ausgabe-Erweiterungs-ROM's abgestellt (siehe Seite 88) und vom Lautsprecher ein kurzer Ton gegeben.

Diese Vorgänge werden jedesmal ausgeführt, wenn Sie die RESET -Taste auf Ihrem Apple betätigen und wieder loslassen. An dieser Stelle sieht nun der Autostart-ROM an zwei bestimmten Speicherstellen nach, ob schon vorher ein RESET gegeben wurde, oder ob der Apple gerade erst angestellt wurde. So kann dann ein „Warm"- oder ein „Kalt"-Start durchgeführt werden. (Diese speziellen Speicherstellen werden weiter unten beschrieben.)

1. **Kaltstart.** Der RESET -Vorgang eines frisch aktivierten Apple löscht nun den Bildschirm und schreibt „Apple II" in der Mitte oben. Dann werden die oben genannten Speicherstellen so gesetzt, daß der Apple später weiß, daß er schon eingeschaltet war. Daraufhin sucht der Apple eine Disk II Steuerkarte (Disk II Controller Card) in den Geräteanschlüssen 7 bis 1 (in dieser Reihenfolge). Findet er eine, so wird das Apple Disk Operating System (Disk Steuerprogramm) aus der Diskette im angeschlossenen Laufwerk geladen (Eine Beschreibung der Initialisierung finden Sie im DOS-Handbuch auf Seite 9).

   Wenn der Autostart-ROM keine Disk Controller Card findet oder Sie nocheinmal RESET betätigen, bevor das Disk Operating System geladen ist, wird mit einem „lauwarmen Start" die in Ihrem apple im ROM vorhandene Sprache neu gestartet. Diese Initialisierung löscht eventuell vorhandene Variablen und Programme. Bei Apples mit der Version Ø-Platine ohne Applesoft II Firmware Karte oder mit einer solchen Karten und dem Kontrollschalter in der DOWN-Position startet der Autostart-ROM das Apple Integer BASIC.

   Apple II Plus-Systeme oder Apple II-Systeme der Version Ø mit der Applesoft II Firmware Karte mit dem Kontrollschalter auf „UP" beginnen in Apple II Fließkomma-BASIC (Applesoft).

2. **Warmstart.** Wenn Sie einen Autostart-ROM haben, der schon einen Kaltstart ausgeführt hat, werden Sie jedesmal, wenn Sie die RESET -Taste drücken und wieder loslassen, in Ihre Sprache zurückkehren und Ihr Programm und Ihre Variablen unversehrt vorfinden.

* Der Einschalt-RESET-Ablauf findet nur auf Apples der Version 1 oder auf Apples der Version Ø mit mindestens einer Disk II Controller Card statt.

Spezielle Speicherstellen des Autostart-ROM's

Die drei speziellen Speicherstellen für den **RESET** -Ablauf des Autostart Monitor-ROM's liegen in einem RAM-Bereich, der für solche Systemfunktionen reserviert ist.

Tabelle 13: Spezielle Adressen für den Autostart-ROM

Adresse Dezimal	Hexa	Inhalt
1010 1011	$3F2 $3F3	Eingangsadresse. Diese zwei Speicherstellen ent-enthalten die Adresse für den Sprung in die Sprache, die gerade benutzt wird (Normalerweise ist es $EØØ3).
1012	$3F4	Einschalt-Byte. (Normalerweise ist es $45) Siehe unten.
64367 -1169	$FB6F	Dies ist die Startadresse eines Unterprogramms in Maschinensprache, das das Einschaltbyte initialisiert.

Wenn der Apple eingeschaltet wird, setzt der Apple das Einschalt-Byte auf einen bestimmten Wert. Dieser Wert bildet sich durch ein Exclusive-OR (Ausschließendes ODER) aus der konstanten Zahl 165 mit dem Inhalt der Speicherstelle 1Ø11. Enthält die Zelle 1Ø11 den Wert 224 (den normalen Wert), so ergibt sich für das Einschalt-Byte:

	Dezimal	Hexa	Binär
Speicherstelle 1011	224	$EØ	111ØØØØØ
Konstante Zahl	165	$A5	1Ø1ØØ1Ø1
Einschalt-Byte	69	$45	Ø1ØØØ1Ø1

Ihr Programm kann die Eingangsadresse ändern, damit bei einem **RESET** in ein anderes Programm oder in eine andere Sprache gesprungen wird. Wenn Sie die Eingangsadresse ändern, sollten Sie sich vergewissern, daß sich der Wert des Einschalt-Bytes aus einem Exclusive-OR von 165 mit dem Inhalt der Speicherstelle 1Ø11 ergibt. Setzen Sie nicht das Einschaltbyte, so wird der Autostart-ROM beim nächsten **RESET** annehmen, daß gerade eingeschaltet wurde und einen Kaltstart durchführen.

So können Sie z.B. die Eingangsadresse auf den Apple System-Monitor richten, damit Sie durch ein **RESET** in den Monitor kommen. Dazu müssen Sie die Startadresse des Monitors in die Speicherstellen der Eingangsadresse eintragen. Der Monitor beginnt ab Adresse $FF69 oder dezimal 65385. Speichern Sie die beiden letzten hexadezimalen Ziffern der Adresse ($69) nach $3F2 und die ersten zwei Hexadezimalziffern ($FF) nach $3F3. Arbeiten Sie dezimal, so speichern Sie 255 (das ist der ganzzahlige Anteil aus 65385/255) nach 1Ø11 und 1Ø5 (das ist der Rest aus dieser Division) nach 1Ø1Ø.

Nun muß das Einschalt-Byte gesetzt werden. Dazu gibt es ein Unterprogramm in Maschinensprache im Autostart-ROM, das automatisch das Ergebnis aus der oben erwähnten Exclusiven ODER-Funktion einsetzt. Sie brauchen also nur ein JSR (Sprung ins Unterprogramm) zu der Adresse $FB6F ausführen. Arbeiten Sie in BASIC, so führen Sie ein CALL-1169 aus. Nun ist alles gesetzt, damit Sie beim nächsten RESET in den Monitor kommen.

Wollen Sie die RESET -Taste wieder normal arbeiten lassen, brauchen Sie nur die alten Werte der Eingangsadresse wiederherzustellen ($E003 oder dezimal 57347) und dieses Unterprogramm ab $FB6F zu starten.

Der Reset des „Alten Monitor"-ROM

Ein RESET -Ablauf im Apple II Monitor-ROM beginnt mit dem Setzen des „Normal"-Modus und dem Einstellen der ersten Seite Test mit gemischter Lo-Res Graphik dahinter. Dann wird das Textfenster voll geöffnet und das Tastenfeld als Eingabegerät und der Bildschirm als Ausgabegerät bestimmt. Der Lautsprecher gibt einen Ton ab und es erscheint das Bereitschaftszeichen (*) des System-Monitors, der eine Eingabe von Ihnen erwartet.

KAPITEL 3
Der System-Monitor

Tief verborgen im Inneren des Apple ROM befindet sich ein meisterhaftes Programm: der System-Monitor. Er ist Überwacher und trotzdem Sklave des Systems. Er kontrolliert alle Programme und alle Programme brauchen ihn. Sie können die gewaltigen Einrichtungen des Monitors ausnutzen, um den versteckten Geheimnissen in allen 65536 Speicherstellen auf die Spur zu kommen. Vom Monitor aus sind Sie in der Lage, sich ein, zwei oder auch alle Adressen anzusehen und deren Inhalte zu verändern. Sie können Programme in Maschinen- oder in Assemblersprache schreiben und sie direkt vom Mikroprozessor ausführen lassen. Sie können unermeßliche Mengen von Daten und Programmen auf Kassette schreiben und alles wieder einlesen. Mit einem einzigen Kommando können Sie Tausende von Bytes vergleichen oder transportieren und Sie können den Monitor verlassen und irgendein anderes Programm oder in einer anderen Sprache des Apple beginnen.

Einstieg in den Monitor

Das Programm des Apple System-Monitors beginnt ab der Adresse $FF69 (dezimal 65385 oder –151). Den Monitor können Sie oder Ihr BASIC-Programm mit CALL –151 aufrufen. Das Bereitschaftszeichen des Monitors, ein Stern (*), erscheint links auf dem Bildschirm und zu seiner Rechten ein blinkender Zeiger. Der Monitor akzeptiert Standard-Eingabezeilen wie alle anderen Systeme und Sprachen auf dem Apple und unternimmt nichts, bis Sie **RETURN** gegeben haben. Ihre Eingabezeilen dürfen für den Monitor bis zu 255 Zeichen lang sein. Sie können Ihren Aufenthalt im Monitor beenden und in die Sprache zurückkehren, die Sie zuvor benutzt haben, indem Sie **CTRL C** **RETURN** geben (oder mit dem Apple DOS: **3** **D** **0** **G** **RETURN**) oder einfach durch **RESET***.

Adressen und Daten

Sie sprechen zu dem Monitor, wie Sie in den meisten Programmen und Sprachen sprechen: Sie tippen eine Zeile auf der Tastatur und dann ein **RETURN**. Nun wird der Monitor das verarbeiten, was Sie ihm eingegeben haben und diesen Anweisungen gemäß handeln. Sie werden dem Monitor drei Arten von Informationen geben: Adressen, Werte und Kommandos (Befehle).

Adressen und Werte nimmt der Monitor nur in hexadezimaler Schreibweise an. Diese Schreibweise verwendet die zehn dezimalen Ziffern 0 bis 9 (für die Werte 0 bis 9) und die Buchstaben A bis F (um die Zahlen 10 bis 15 darzustellen). Eine Hexadezimalziffer kann deshalb Werte von 0 bis 15 annehmen. Zwei Hexadezimalziffern stellen Zahlen zwischen 0 und 255 dar und eine Gruppe von vier Ziffern überstreicht den Bereich von 0 bis 65535. Also wird jede Adresse im Apple durch vier Hexadezimalziffern dargestellt und jeder Wert (Inhalt einer Speicherstelle) durch zwei Hexadezimalziffern. So sprechen Sie mit dem Monitor über Adressen und Werte. Wenn der Monitor eine Adresse haben will, akzeptiert er jede Gruppe von Hexadezimalziffern. Sind weniger als vier Ziffern in dieser Gruppe, so wird er führende Nullen ergänzen, gibt es mehr als vier Ziffern, so werden nur die letzten vier Ziffern ausgewertet. Entsprechend behandelt der Monitor die Eingabe von zweiziffrigen Datenwerten.

Der Monitor kennt 22 verschiedene Kommandozeichen. Einige sind Satzzeichen, andere sind Großbuchstaben oder Steuerzeichen. In den folgenden Abschnitten wird der volle Name eines Kommandos in Großbuchstaben erscheinen. Der Monitor braucht nur den ersten Buchstaben des Kommandonamens. Einige Kommandos werden durch Steuerzeichen aufgerufen. Obwohl der Monitor diese Zeichen erkennt und interpretiert, wird ein Steuerzeichen einer Eingabezeile nicht auf dem Billdschirm erscheinen.

---

* Dies funktioniert nicht auf Apples ohne Autostart-ROM.

Prüfen der Inhalte einzelner Speicherstellen

In den folgenden Beispielen werden zur Verdeutlichung Ihre Eingaben *buchstabenweise unterstrichen*, um sie von den Antworten des Apple zu unterscheiden. Die Werte in diesen Beispielen können von den Werten abweichen, die Sie auf Ihrem Apple bei denselben Anweisungen erhalten.

Wenn Sie nur die Adresse einer Speicherstelle eingeben, wird der Monitor so antworten: Die Adresse, die Sie eingegeben haben, ein Gedankenstrich, ein Leerzeichen und der Wert in dieser Speicherzelle.

```
*E000

E000-  20
*300

0300-  99
*
```

Untersuchung weiterer Speicherstellen

Geben Sie dem Monitor in einer Eingabezeile einen Punkt (.) und dann eine Adresse, so wird ein „Speicherauszug" ausgegeben: Es sind die Werte aller Speicherstellen von der zuletzt geöffneten Speicherstelle bis zu der nach dem Punkt angegebenen Adresse. Die zuletzt angezeigte Speicherstelle wird wieder als die zuletzt geöffnete und als die nächste veränderbare Speicherstelle betrachtet

```
*20

0020-  00
*.2B

0021-  28  00  18  0F  0C  00  00
0028-  A8  06  D0  07
*300

0300-  99
*.315

030-   B9  00  08  0A  0A  0A  99
0308-  00  08  C8  D0  F4  A6  2B  A9
0310-  09  85  27  AD  CC  03
*.32A

0316-  85  41
0318-  84  40  8A  4A  4A  4A  4A  09
0320-  C0  85  3F  A9  5D  85  3E  20
0328-  43  03  20
*
```

Hier sind noch einige Bemerkungen zum Format eines Speicherauszugs. Erstens: Die erste Zeile eines *Speicherauszugs beginnt mit der Adresse, die der zuletzt geöffneten Speicherstelle folgt.* Zweitens: Die anderen Zeilen beginnen mit Adressen, die mit einer Acht ($...8) enden oder mit einer Null ($...0). Schließlich werden nicht mehr als acht Werte in einer Zeile ausgegeben. Wenn der Monitor einen Speicherauszug ausgibt, startet er mit der Anzeige der Adresse und des Wertes der Speicherzelle, die der zuletzt geöffneten folgt. Dann wird die nächste Speicherstelle bearbeitet.

Falls die Adresse dieser Speicherstelle mit einer 8 oder einer 0 endet, „schneidet" er die Zeile „ab" und bringt diese Adresse auf eine neue Zeile, um dort mit der Ausgabe der Werte fortzufahren. Nach der Anzeige des Speicherinhalts der Zelle, deren Adresse Sie verlangt haben, wird der Speicherauszug beendet. Wieder ergibt sich die zuletzt ausgegebene Adresse als die zuletzt geöffnete und als die nächste veränderbare Adresse. Ist die angegebene Adresse in einer Eingabezeile kleiner als die zuletzt geöffnete Adresse, so gibt der Monitor nur die Speicheradresse mit ihrem Inhalt aus, die der zuletzt geöffneten folgt.

Sie können die zwei Kommandos (Öffnen und Speicherauszug) auf einmal eingeben: tippen Sie die Anfangsadresse, dann einen Punkt und die Endadresse. Diese zwei Adressen, die durch einen Punkt getrennt sind, nennt man „Speicherbereich".

```
*300.32F

0300- 99 B9 00 08 0A 0A 0A 99
0308- 00 08 C8 D0 F4 A6 2B A9
0310- 09 85 27 AD CC 03 85 41
0318- 84 40 8A 4A 4A 4A 4A 09
0320- C0 85 3F A9 5D 85 3E 20
0328- 43 03 20 46 03 A5 3D 4D
*30.40

0030- AA 00 FF AA 05 C2 05 C2
0038- 1B FD D0 03 3C 00 40 00
0040- 30
E015.E025

E015- 4C ED FD
E018- A9 20 C5 24 B0 0C A9 8D
E020- A0 07 20 ED FD A9
*
```

Und noch mehr Speicherstellen

Ein einfacher Druck der RETURN-Taste veranlaßt den Monitor, mit einer Speicherauszugszeile zu antworten. Die Speicherausgabe erstreckt sich von der Adresse, die der zuletzt geöffneten Speicherstelle folgt, bis zur nächsten Adresse, die mit einer 7 ($...7) oder mit einem F ($...F) endet. Wieder wird die zuletzt angezeigte Adresse als zuletzt geöffnet und als nächste Veränderbare betrachtet.

```
*5

0005-  00
*RETURN
 00 00
*RETURN

0008-  00  00  00  00  00  00  00  00
*32

0032-  FF
*RETURN
 AA 00 C2 05 C2
*RETURN

0038-  1B  FD  D0  03  3C  00  3F  00
*
```

Ändern des Inhalts einer Speicherzelle

Sie haben bereits alles über die „nächste veränderbare Speicherstelle" erfahren. Damit Sie auch sehen, was wirklich passiert, tippen Sie einen Doppelpunkt und dann einen Wert.

```
*0

0000-  00
*:5F
```

Gut!
Sehen Sie schnell nach, denn der Wert dieser Speicherstelle hat sich tatsächlich sofort verändert und Ihren Eingabewert angenommen:

```
*0

0000-  5F
*
```

Sie können das Öffnen und Ändern auch zu einer Anweisung zusammenfassen:.

```
*302:42

*302

0302-  42
*
```

Wenn Sie den Inhalt einer Speicherstelle ändern, verschwindet der alte Wert, den diese Speicherzelle hielt, und kann nie wieder ermittelt werden.
Der neue Wert steht dann in der Speicherzelle bis auch er von einem anderen Wert überschrieben wird.

Ändern der Inhalte von aufeinderfolgenden Speicherstellen

Sie brauchen nicht für jede Speicherstelle immer eine Adresse, einen Doppelpunkt, einen Wert und ein RETURN eingeben, wenn Sie mehrere Adressen hintereinander ändern wollen. Der Monitor macht es Ihnen möglich, die Werte von maximal achtundfünfzig Speicherstellen auf einmal zu ändern. Dazu geben Sie die Anfangsadresse, einen Doppelpunkt und dann alle Werte ein, die durch Leerzeichen voneinander getrennt sein müssen. Der Monitor trägt nun ab der Anfangsadresse hintereinander die Werte in aufeinanderfolgende Speicherzellen ein. Ist er damit fertig, so bestimmt er die Adresse nach der zuletzt geänderten Speicherstelle als die „nächste veränderbare Adresse". Wollen Sie nun noch mehr Speicherstellen hinter der zuletzt veränderten Speicherzelle ändern, so brauchen Sie die Anfangsadresse nicht anzugeben. Die nächste Eintragung neuer Werte wird automatisch nach der letzten geänderten Speicherstelle fortgesetzt. Sie geben dann nur einen Doppelpunkt und weitere Werte ein.

```
*300:69 01 20 ED  FD 4C 0 3

*300

0300- 69
*RETURN
 01 20 ED FD 4C 00 03
*10:0 1 2 3

*:4 5 6 7

*10.17

0010- 00  01  02  03  04  05  06  07
*
```

Verlegen eines Speicherbereichs

Sie können die Inhalte eines Speicherbereichs (festgelegt durch zwei mit einem Punkt voneinander getrennte Speicheradressen) als ein Ganzes auffassen und sie dann mit dem MOVE-Kommando des Monitors von einer Stelle zu einer anderen bringen. Dazu muß dem Monitor angegeben werden, wo der Speicherbereich liegt und wo er hin soll.

Diese Information besteht aus folgenden Teilen: der Zieladresse, einer linken spitzen Klammer (<), der Anfangs- und der Endadresse des Bereichs und einem „M", damit der Monitor einen Transport (Move) durchführt. Die Anfangs- und die Endadresse geben Sie in gewohnter Weise an (durch einen Punkt getrennt). Das Kommando sieht dann so aus:

{Ziel} < {Anfang} . {Ende} M

Wenn Sie diese Zeile tippen, sollten Sie die Worte in geschweiften durch hexadezimale Adressen ersetzen. Hier sind ein paar Beispiele:

```
*0.F

0000- 5F  00  05  07  00  00  00  00
0008- 00  00  00  00  00  00  00  00
*300:A9 8D 20 ED FD A9 45 20 DA FD 4C 00 03

*300.30C

0300- A9  8D  20  ED  FD  A9  45  20
0308- DA  FD  4C  00  03
*0<300.30CM

*0.C

0000- A9  8D  20  ED  FD  A9  45  20
0008- DA  FD  4C  00  03
*310<8.AM

*310.312

0310- DA  FD  4C
* 2<7.9M

*0.C

0000- A9  8D  20  DA  FD  A9  45  20
0008- DA  FD  4C  00  03
*
```

Der Monitor erstellt einfach eine Kopie des angegebenen Bereichs und bringt sie an den angegebenen Bestimmungsort. Der „Original"-Bereich bleibt unverändert. Die Endadresse des Orginalbereichs wird jetzt die zuletzt geöffnete Adresse und die nächste veränderbare Adresse ergibt sich aus der Anfangsadresse des Orginalbereichs. Ist die zweite Adresse des Bereichs kleiner als die erste, so wird nur ein Wert (nämlich der Wert der ersten Speicherstelle des Bereichs) transportiert.

Liegt die Zieladresse des MOVE-Kommandos innerhalb des Orginalbereichs, so geschehen merkwürdige und (manchmal) wunderbare Dinge:

Die Adressen zwischen dem Anfang des Bereichs und der Zieladresse werden als Teilbereiche behandelt, deren Werte mehrfach in den Originalbereich kopiert werden, bis sie diesen ausfüllen (siehe Seite 59 : „Besondere Kniffe").

Vergleichen von zwei Speicherbereichen

Sie können mit dem Monitor zwei Speicherbereiche miteinander vergleichen. Dazu verwenden Sie dasselbe Format, wie Sie es schon vom MOVE-Kommando herkennen. So läßt sich mit dem VERIFY-Kommando direkt nach einem MOVE-Kommando feststellen, ob der Transport geglückt ist.

Das VERIFY-Kommando braucht wie das MOVE-Kommando eine Zielangabe und einen Bereich :

{Ziel} < {Anfang} . {Ende} V

Der Monitor vergleicht den angegebenen Bereich mit dem Bereich ab der Zielangabe. Bestehen irgendwelche Unterschiede, so gibt der Monitor die Adressen mit den jeweiligen unterschiedlichen Werten aus.

```
*0:D7 F2 E9 F4 F4 E5 EE A0 E2 F9 A0 C3 C4 C5

*300<0.DM

*300<0.DV

*6:E4

*300<0.DV

0006- E4  (EE)
*
```

Beachten Sie, daß das VERIFY-Kommando die Adresse derjenigen Speicherstelle im Orginalbereich angibt, deren Wert nicht mit dem Wert des entsprechenden Bereichs übereinstimmt. Gibt es keine Unterschiede, so gibt VERIFY nichts aus. Beide Bereiche bleiben unverändert. Die letzte geöffnete und die nächste veränderbare Adresse ergibt sich jeweils wie im MOVE-Kommando. Entsprechend werden auch nur die Anfangsadressen verglichen, wenn die Endadresse kleiner ist als die Anfangsadresse. Auch VERIFY arbeitet ungewöhnlich, wenn das Ziel im Orginalbereich liegt (siehe Seite 68: Besondere Kniffe).

Schreiben eines Speicherbereichs auf Band

Der Monitor hat zwei spezielle Kommandos, die Ihnen erlauben, Speicherbereiche auf Kassette zu schreiben und für den späteren Gebrauch wieder einzulesen. Das erste dieser beiden Kommandos, das WRITE-Kommando, schreibt Ihnen die Inhalte von einer oder bis zu 65536 Speicherstellen auf Standard-Kassetten.

Um nun solch einen Speicherbereich auf Kassette zu schreiben, geben Sie dem Monitor die Anfangs- und die Endadresse des Bereichs ein, gefolgt von einem „W" (für WRITE = Schreiben).

{Anfang} . {Ende} W

Um fehlerfrei aufnehmen zu können, sollten Sie den Kassettenrekorder auf „Aufnahme" stellen, bevor Sie RETURN auf der Eingabezeile geben. Lassen Sie das Band ein paar Sekunden laufen und geben Sie dann RETURN. Der Monitor wird eine 10 Sekunden lange Kopfinformation auf Band schreiben und dann erst die Daten. Ist der Monitor fertig, so meldet er sich mit einem kurzen Ton und wartet auf weitere Anweisungen. Es empfiehlt sich dann, das Band zurückzuspulen und es mit verständlichen Angaben über den abgespeicherten Inhalt und seine Aufgabe zu versehen.

```
*0:FF FF AD 30 C0 88 D0 04 C6 01 F0 08 CA D0 F6 A6 00 4C 02 00 60

*0.14
```

```
0000- FF FF AD 30 C0 88 D0 04
0008- C6 01 F0 08 CA D0 F6 A6
0010- 00 4C 02 00 60
*0.14W

*
```

Es dauert etwa 35 Sekunden, um nach der 10 Sekunden-Kopfinformation die Werte von 4096 Speicherstellen auf Band zu schreiben. Es werden etwa 1350 Bit pro Sekunde übertragen. Wenn alle Werte aufgezeichnet sind, schreibt der Monitor noch einen zusätzlichen Wert auf das Band. Es ist die „Prüfsumme", die aus allen Werten des übertragenen Speicherbereichs gebildet wird. Das READ-Kommando (siehe unten) benutzt dann diesen Wert, um Übertragungsfehler festzustellen. (Wen es interessiert: Die Prüfsumme ist anfangs $FF und wird durch Exclusive-OR von jedem Wert des übertragenen Bereichs verändert.)

Einlesen eines Speicherbereichs vom Band

Haben Sie mit dem WRITE-Kommando des Monitors einen Speicherbereich auf Band geschrieben, so können Sie ihn mit dem READ-Kommando des Monitors wieder einlesen. Es muß nicht derselbe Speicherbereich sein, der die Daten vom Band bekommt. Sie können den Monitor auch anweisen, diese Daten in einen anderen Speicherbereich der gleichen Größe zu geben.

Das Format des READ-Kommandos ist dasselbe wie bei WRITE, nur steht ein „R" an der Stelle des Kommandobuchstabens:

{Anfang} . {Ende} R

Geben Sie auch hier nicht sofort das **RETURN**, sondern stellen Sie den Kassettenrekorder auf „Wiedergabe" und warten Sie, bis das Vorspulband durchgelaufen ist. Obwohl das WRITE-Kommando eine 10 Sekunden lange Kopfinformation geschrieben hat, braucht das READ-Kommando nur drei Sekunden, um sich auf die Frequenz einzustellen. So sollten Sie ein paar Sekunden vergehen lassen, bis Sie **RETURN** eingeben, damit der Kassettenausgang einen gleichmäßigen Ton abgeben kann.

```
*0:0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


*0.14

0000- 00 00 00 00 00 00 00 00
0008- 00 00 00 00 00 00 00 00
0010- 00 00 00 00 00
*0.14R

*0.14


0000- FF FF AD 30 C0 88 D0 04
0008- C6 01 F0 08 CA D0 F6 A6
0010- 00 4C 02 00 60
*
```

Nachdem der Monitor alle Werte gelesen und gespeichert hat, liest er den Wert der Prüfsumme und vergleicht ihn mit der eigenen Prüfsumme. Weichen die beiden Werte voneinander ab, dann gibt der Monitor einen Ton von sich und schreibt „ERR" auf den Bildschirm. So sind Sie gewarnt, daß beim Einlesen Probleme auftraten und die Werte nicht mit denen übereinstimmen, die Sie aufs Band geschrieben haben. Stimmt aber die Prüfsumme, so erwartet der Monitor weitere Anweisungen von Ihnen.

Erstellen und starten von Maschinenprogrammen

Die Maschinensprache ist sicher die wirksamste Sprache auf dem Apple, obgleich auch die unbequemste in der Handhabung. Der Monitor hat spezielle Einrichtungen, um den Programmierern, die sich mit Maschinensprache befassen, beim Erstellen, Eintragen und Korrigieren und nicht zuletzt bei der Fehlersuche zu helfen.

Sie können ein Maschinenprogramm schreiben und die hexadezimalen Werte der Befehlsteile und der zugehörigen Adressteile mit den oben beschriebenen Monitorkommandos in die Speicherzellen eintragen. Mit Hilfe des Monitors können Sie einen hexadezimalen Speicherauszug Ihres Programms erhalten, es überall im Speicher herumtransportieren oder es auf Band schreiben und es wieder einlesen. Das wichtigste Kommando im Zusammenhang mit Maschinensprache ist aber das GO-Kommando. Wenn Sie eine Speicherstelle öffnen und den Buchstaben „G" tippen, wird der Monitor den Mikroprozessor veranlassen, an der geöffneten Adresse mit der Ausführung des Maschinenprogramms zu beginnen. Der Monitor behandelt dieses Programm wie ein Unterprogramm: am Ende der Ausführungen sollte ein RTS-Befehl (Rücksprung aus dem Unterprogramm) stehen, um die Kontrolle wieder an den Monitor zu geben.
Ihre Programme in Maschinensprache können viele Unterprogramme des Monitors aufrufen. Hier wird ein Programm, das die Buchstaben „A" bis „Z" ausgibt, eingegeben und gestartet:

```
*300:A9 C1 20 ED FD 18 69 1 C9 DB D0 F6 60

*300.30C

0300-  A9  C1  20  ED  FD  18  69  01
0308-  C9  DB  D0  F6  60
*300G
ABCDEFGHIJKLMNOPQRSTUVWXYZ
*
```

(Den Befehlssatz des 6502-Mikroprozessors finden Sie im Anhang A dieses Handbuchs.)

Ein hexadezimaler Speicherauszug eignet sich nicht gerade besonders gut, um Programme in Maschinensprache zu lesen und vielleicht dabei auch noch Fehler zu suchen. Also muß es etwas anderes geben, um sich Maschinenprogramme anzusehen.

Im Monitor des Apple gibt es nun ein Kommando, das Maschinenprogramme in Assemblersprache ausgibt. Das bedeutet, daß eine unformatierte Menge von Hexadezimalziffern in einzelne Befehle von 1, 2 oder 3 Byte zerlegt wird. Das LIST-Kommando des Monitors gibt nun ab der angegebenen Speicherstelle einen Bildschirm (20 Zeilen) voller Befehle aus:

```
*300L

0300-   A9 C1      LDA   #$C1
0302-   20 ED FD   JSR   $FDED
0305-   18         CLC
0306-   69 01      ADC   #$01
0308-   C9 DB      CMP   #$DB
030A-   D0 F6      BNE   $0302
030C-   60         RTS
030D-   00         BRK
030E-   00         BRK
030F-   00         BRK
0310-   00         BRK
0311-   00         BRK
0312-   00         BRK
0313-   00         BRK
0314-   00         BRK
0315-   00         BRK
0316-   00         BRK
0317-   00         BRK
0318-   00         BRK
0319-   00         BRK
*
```

Erkennen Sie die ersten Zeilen? Es ist das Maschinenprogramm, das eben noch die Buchstaben „A" bis „Z" ausgegeben hatte, nur sind diesmal in der Assemblerform die einzelnen Befehle zum Vorschein gekommen. Die restlichen Befehle (BRK-Befehle) erscheinen, weil immer 20 Zeilen ausgegeben werden. Ihre angegebene Adresse hat sich der Monitor gemerkt, aber nicht in der gewohnten Weise. Sie wurde in den „Befehlszähler" gegeben, der nur dazu da ist, auf Speicherzellen des Programms zu zeigen. Nach einem LIST-Kommando weist er auf die Adresse, die der zuletzt auf dem Bildschirm angezeigten Adresse folgt. Wenn Sie jetzt ein weiteres LIST geben, wird es wieder einen Bildschirm voller Befehle ausgeben, die sich an die Befehle des ersten Bildschirms anschließen.

Der Mini-Assembler

Es gibt ein weiteres Programm im Monitor*, das Ihnen ermöglicht, Programme im selben Assemblerformat einzugeben, wie es das LIST-Kommando ausgibt. Dies Programm ist der Apple-Mini-Assembler. Es ist „Mini", weil es nicht symbolische Adressen verarbeiten kann, was voll entwickelte Assembler können müssen. So starten Sie den Mini-Assembler:

```
*F666G

!
```

Nun sind Sie im Mini-Assembler. Das Ausrufezeichen ist sein Bereitschaftszeichen. Während Sie im Mini-Assembler sind, können Sie jedes Monitor-Kommando geben, wenn Sie ein „$" voranstellen. Ansonsten hat der Mini-Assembler seine eigene Syntax und seine eigenen Kommandos.

* Der Mini-Assembler befindet sich eigentlich nicht im ROM des Monitors, ist aber im INTEGER BASIC-ROM enthalten. Daher ist er auch nicht in Apple II Plus-Systemen bzw. beim Gebrauch der Firmware Applesoft II verfügbar.

Der Mini-Assembler kennt den Befehlszähler des Monitors. Bevor Sie ein Programm eingeben, müssen Sie den Befehlszähler auf die Adresse setzen, wo Ihr Programm abgelegt werden soll. Geben Sie dazu diese Adresse und einen Doppelpunkt ein. Dann folgt der erste Befehl Ihres Programms mit den 3 Buchstaben des Befehlssymbols und einem Leerzeichen. Nun geben Sie den Operanden dieses Befehls ein und dann das RETURN. Der Mini-Assembler formt diese Zeile zu Hexadezimalzahlen um und speichert sie ab, beginnend mit der Adresse, auf die der Befehlzähler zeigt. Diese (höchstens 3) Hexadezimalzahlen werden nun wie beim LIST-Kommando des Monitors wieder ins Assemblerformat gebracht und auf den Bildschirm gegeben. (Damit sehen Sie noch einmal ausführlich Ihre eingetippte Zeile). Dann meldet sich der Mini-Assembler wieder mit einem Ausrufezeichen. Sollen die Befehle aufeinander folgen, so braucht bei der Eingabe nicht immer erst die Adresse und der Doppelpunkt gegeben zu werden. Tippen Sie also einfach ein Leerzeichen, den Befehl (Befehlssymbol, Leerzeichen, Operand) und ein RETURN. Diese Zeile wird übersetzt und schon können Sie die nächste eingeben.

Falls die eingegebene Zeile einen Fehler in sich birgt, meldet sich der Mini-Assembler mit einem Klingelzeichen und bringt ein Zirkumflex (^) unter das falsche Zeichen (oder zumindest in die Nähe). Meist sind es Tippfehler: falsche Befehlssymbole, fehlende Klammern, u.s.w. Der Mini-Assembler lehnt auch Eingabezeilen ab, wenn Sie das Leerzeichen vor oder hinter dem Befehlssymbol vergessen, oder ein fremdes Zeichen in hexadezimale Werte oder Adressen bringen. Ist die Zieladresse eines relativen Sprungs zu weit vom Sprungsbefehl entfernt (mehr als 127 Adressen Entfernung von der Adresse des Sprungbefehls), so zeigt der Mini-Assembler dies auch als Fehler an.

```
!300:LDX =02

0300-    A2 02       LDX    #$02
! LDA $0,X

0302-    B5 00       LDA    $00,X
! STA $10,X

0304-    95 10       STA    $10,X
! DEX

0306-    CA          DEX
! STA $C030

0307-    8D 30 C0    STA    $C030
! BPL $302

030A-    10 F6       BPL    $0302
! BRK

030C-    00          BRK
!
```

Sie verlassen den Mini-Assembler und kommen in den Monitor, wenn Sie RESET geben oder das Monitorkommando (eingeleitet durch das $-Zeichen) :

```
!$FF69G

*
```

Ihr Programm in Assemblersprache ist nun gespeichert. Sie können es sich wieder mit dem LIST-Kommando ansehen :

```
*300L

0300-   A2 02      LDX   #$02
0302-   B5 00      LDA   $00,X
0304-   95 00      STA   $10,X
0306-   CA         DEX
0307-   8D 30 C0   STA   $C030
030A-   10 F6      BPL   $0302
030C-   00         BRK
030D-   00         BRK
030E-   00         BRK
030F-   00         BRK
0310-   00         BRK
0311-   00         BRK
0312-   00         BRK
0313-   00         BRK
0314-   00         BRK
0315-   00         BRK
0316-   00         BRK
0317-   00         BRK
0318-   00         BRK
0319-   00         BRK
*
```

Fehlersuche

Für das Verständnis von Maschinenprogrammen ist es sehr vorteilhaft, wenn man ihren Ablauf in „Zeitlupe" betrachten kann. Dazu gibt es im Monitor ein STEP-Kommando*, das schrittweise Maschinenprogramme auswertet, ausführt und anzeigt. Außerdem gibt es ein TRACE-Kommando*, das diese schrittweise Handlung automatisch durchführt und erst anhält, wenn ein BRK-Befehl erreicht wird. Mit diesen beiden Monitor-Kommandos können Sie in Ihren Maschinen- oder Assemblerprogrammen auch den letzten Fehler ausfindig machen.

Jedes STEP-Kommando (step = Schritt) läßt den Monitor den Befehl ausführen, auf den der Befehlszähler gerade zeigt. Der Befehl wird wie im LIST-Kommando als Assemblerzeile gedruckt und dann ausgeführt. Danach werden die Inhalte der internen Register des 6502-Mikroprozessors ausgegeben. Schließlich zeigt der Befehlszähler dann auf den nächsten Befehl des Maschinenprogramms.

Hier sehen Sie, was passiert, wenn Sie mit STEP das eingegebene Mini-Assemblerprogramm schrittweise ausführen lassen:

```
*300S

0300-   A2 02      LDX   #$02
 A=0A X=02 Y=D8 P=30 S=F8
*S
```

* Die STEP- und TRACE-Kommandos sind nicht auf Apples mit dem Autostart-ROM verfügbar.

```
0302-    B5 00        LDA    $00,X
 A=0C X=02 Y=D8 P=30 S=F8
*S

0304-    95 10        STA    $10,X
 A=0C X=02 Y=D8 P=30 S=F8
*12

0012-    0C
*S

0306-    CA           DEX
 A=0C X=01 Y=D8 P=30 S=F8
*S

0307-    8D 30 C0     STA    $C030
 A=0C X=01 Y=D8 P=30 S=F8
*S

030A-    10 F6        BPL    $0302
 A=0C X=01 Y=D8 P=30 S=F8
*S

0302-    B5 00        LDA    $00,X
 A=0B X=02 Y=D8 P=30 S=F8
*S

0304-    95 10        STA    $10,X
 A=0B X=01 Y=D8 P=30 S=F8
*
```

Beachten Sie, daß nach der Ausführung des dritten Befehls der Inhalt der Speicherstelle $12 geprüft wurde. Es ergab sich der erwartete Wert und die schrittweise Arbeitsweise konnte fortgesetzt werden. Der Monitor hält die zuletzt geöffnete Adresse und den Befehlszähler voneinander getrennt. So können Sie während der schrittweisen Ausführung Ihres Programms die Speicherstellen kontrollieren oder ändern.

Das TRACE-Kommando (trace = Spur) ist nichts anderes als ein ständig wiederholtes STEP-Kommando. Es stoppt erst die Ausführung eines Programms, wenn Sie ein **RESET** geben oder ein BRK-Befehl in Ihrem Programm erreicht wird. Trifft TRACE am Ende Ihres Programms auf einen RETURN-Befehl, so wird TRACE seine Tätigkeit irgendwo anders fortsetzen und kann dann nur durch ein **RESET** gestoppt werden.

```
*T

0306-    CA           DEX
 A=0B X=00 Y=D8 P=32 S=F8
0307-    8D 30 C0     STA    $C030
 A=0B X=00 Y=D8 P=32 S=F8
030A-    10 F6        BPL    $0302
 A=0B X=00 Y=D8 P=32 S=F8
0302-    B5 00        LDA    $00,X
 A=0A X=00 Y=D8 P=30 S=F8
0304-    95 10        STA    $10,X
 A=0A X=00 Y=D8 P=30 S=F8
```

```
0306-     CA              DEX
 A=0A X=FF Y=D8 P=B0 S=F8
0307-     8D 30 C0        STA     $C030
 A=0A X=FF Y=D8 P=B0 S=F8
030A-     10 F6           BPL     $0302
 A=0A X=FF Y=D8 P=B0 S=F8
030C-     00              BRK
030C- A=0A X=FF Y=D8 P=B0 S=F8
*
```

Prüfen und ändern von Registerinhalten

Die STEP- und TRACE-Kommandos zeigen nach jedem Programmschritt die Inhalte der internen Register des Mikroprozessors an. Sie können diese Register selbst zur Anzeige bringen oder sie vorbesetzen, wenn Sie ein Maschinenprogramm mit STEP, TRACE oder GO ausführen lassen.

Der Monitor reserviert fünf Speicherstellen für die fünf 6502-Register: A, X, Y, P (Prozessorzustand) und S (Stackpointer = Kellerzeiger). Das EXAMINE-Kommando des Monitors wird durch ein CTRL E ausgelöst und zeigt die Inhalte dieser Adressen an. Die Speicherstelle für das 6502-A-Register ist dann die nächste veränderbare Adresse. Wollen Sie die Werte dieser Speicherstellen ändern, so brauchen Sie nur einen Doppelpunkt und dann die Werte, durch Leerzeichen getrennt, eingeben. Beim nächsten GO, STEP oder TRACE wird der Monitor erst diese Werte in die echten Register des 6502 laden, bevor er den ersten Befehl Ihres Programms ausführen wird.

```
*CTRL E

 A=0A X=FF Y=D8 P=B0 S=F8
*:B0 02

*CTRL E

 A=B0 X=02 Y=D8 P=B0 S=F8
*306S

0306-     CA              DEX
 A=B0 X=01 Y=D8 P=30 S=F8
*S

0307-     8D 30 C0        STA     $C030
 A=B0 X=01 Y=D8 P=30 S=F8
*S

030A-     10 F6           BPL     $0302
 A=B0 X=01 Y=D8 P=30 S=F8

*
```

Verschiedene Monitor-Kommandos

Sie können den Zustand der NORMAL-/INVERS-Darstellung für die Ausgabe durch COUT (siehe Seite 39) vom Monitor aus bestimmen. Das INVERSE-Kommando des Monitors stellt auf inverse Ausgabe um, allerdings bleiben Eingabezeilen in der Normalanzeige. Der NORMAL-Zustand wird dann durch das NORMAL-Kommando des Monitors wieder hergestellt.

```
*0.F

0000- 0A 0B 0C 0D 0E 0F D0 04
0008- C6 01 F0 08 CA D0 F6 A6
*I

*0.F

0000- 0A 0B 0C 0D 0E 0F D0 04
0008- C6 01 F0 08 CA D0 F6 A6
*N

*0.F

0000- 0A 0B 0C 0D 0E 0F D0 04
0008- C6 01 F0 08 CA D0 F6 A6
*
```

Das BASIC-Kommando, durch ein **CTRL B** eingeleitet, gestattet Ihnen, den Monitor zu verlassen und in die Sprache zu kommen, die in ROM's auf Ihrem Apple zur Verfügung steht. Gewöhnlich ist es Apple Integer BASIC oder Applesoft II BASIC. Auf diesem Weg gehen Ihnen aber alle vorhandenen Programme und Variablen verloren. Sind Sie von BASIC in den Monitor gegangen und wollen Sie wieder zurück ins BASIC, ohne Programme und Variablen zu verlieren, so können Sie dass **CTRL C**-Kommando (Continue BASIC) gebrauchen. Ist das Apple Disk Operating System (DOS) aktiv, so kommen Sie durch das «3D0G»-Kommando wieder in die Sprache zurück, in der Sie gerade arbeiten, und werden Ihr Programm und die Variablen unverändert vorfinden.

Das PRINTER-Kommando wird durch ein **CTRL P** gegeben und lenkt alle Ausgaben, die normalerweise für den Bildschirm bestimmt sind, zu einem APPLE INTELLIGENT INTERFACE in einem vorgegebenen Einsteckschlitz. Die Nummer des Einsteckschlitzes sollte 1 bis 7 sein und es sollte dort auch eine Interfacekarte stecken, sonst verlieren Sie die Kontrolle über ihren Apple und damit alle Ihre Programme und Variablen. Das Kommando hat das Format:
{Nummer des Einsteck-Anschlusses} **CTRL P**

Das PRINTER-Kommando «**0 CTRL P**» bringt die folgenden Ausgaben des Apple wieder auf den Bildschirm.

Das KEYBOARD-Kommando ersetzt entsprechend die Tastatur des Apple durch ein anderes Eingabegerät über einen bestimmten Einsteckschlitz. (Weitere Informationen über die Funktion dieser und der entsprechenden BASIC-Kommandos PR# und IN# finden Sie auf Seite 101: «CSW und KSW Sprungadressen».) Das KEYBOARD-Kommando hat das Format:

{Nummer des Einsteck-Anschlusses} **CTRL K**

Ein KEYBOARD-Kommando mit der Anschlußnummer 0 bestimmt die eingebaute Tastatur des Apple als Eingabegerät.

Der Monitor kann auch einfache hexadezimale Additionen und Subtraktionen ausführen. Sie brauchen nur eine Zeile mit diesem Format einzugeben :

{Wert} + {Wert}
{Wert} – {Wert}

Der Apple wird das Ergebnis ausrechnen und anzeigen :

```
*20+13
=33
*4A-C
=3E
*FF+4
=03
*3-4
=FF
*
```

Besondere Kniffe mit dem Monitor

Sie können so viele Monitor-Kommandos in einer Eingabe zusammenfassen, wie Sie wollen, solange Sie sie durch Leerzeichen trennen und die Anzahl aller Zeichen 253 in der Eingabezeile nicht überschreitet. Sie können außer dem STORE-Kommando (:) alle Kommando in beliebiger Reihenfolge angeben. Da der Monitor alle Werte nach dem Doppelpunkt in aufeinanderfolgende Speicherstellen ablegt, muß dem letzten Wert des STORE-Kommandos ein Buchstabenkommando folgen. Das NORMAL-Kommando ist dafür ein gutes Trennzeichen, da es meist keine Veränderung bewirkt und überall verwendet werden kann.

```
*300.307 300:18 69 1 N 300.302 300S S

0300- 00 00 00 00 00 00 00 00
0300- 18 69 01
0300-    18         CLC
 A=04 X=01 Y=D8 P=30 S=F8
0301-    69 01      ADC     #$01
 A=05 X=01 Y=D8 P=30 S=F8
*
```

Kommandos mit einem Buchstaben, wie L, S, I und N brauchen nicht mit Leerzeichen von anderen Kommandos getrennt werden.

Erreicht der Monitor in der Eingabezeile ein Zeichen, das er nicht als Hexadezimalzahl noch als gültiges Kommandozeichen erkennen kann, führt er alle Kommandos bis zu diesem Zeichen aus. Dann meldet er mit einem Tonsignal den Fehler und ignoriert den Rest der Eingabezeile.

Das MOVE-Kommando kann dazu benutzt werden, eine beliebige Folge von Werten in einen Speicherbereich zu übertragen. Dazu wird diese Folge von Werten an den Anfang des Bereichs geschrieben :

```
*300:11 22 33

*
```

Nun kommt es auf die Anzahl der zu wiederholenden Werte an (in diesem Fall sind es drei). Das MOVE-Kommando bekommt dann eine andere Einteilung :
{Anfangsadresse + Anzahl} < {Anfangsadresse} . {Endadresse– Anzahl} M

Dieses MOVE-Kommando kopiert die Folge von Werten hinter die Originalfolge, überträgt diese Kopie in die anschließenden Speicherzellen und wiederholt diesen Vorgang, bis der gesamte Bereich gefüllt ist.

```
*303 300.32DM

*300.32F

0300- 11  22  33  11  22  33  11  22
0308- 33  11  22  33  11  22  33  11
0310- 22  33  11  22  33  11  22  33
0318- 11  22  33  11  22  33  11  22
0320- 33  11  22  33  11  22  33  11
0328- 22  33  11  22  33  11  22  33
*
```

Ein entsprechender Trick kann bei dem VERIFY-Kommando angewandt werden, um zu prüfen, ob sich eine Folge von Werten durch einen ganzen Speicherbereich hindurch wiederholt. Insbesondere läßt sich damit feststellen, ob ein Speicherbereich überall denselben Wert enthält:

```
*300:0

*301 300.31FM

*301 300.31FV

*304:02

*301 300.31FV

0303-00    (02)
0304-02    (00)
*
```

Sie können eine Kommandozeile schreiben, die sich selbst oder einen Teil der Zeile unaufhörlich wiederholt. Dazu fängt der Teil, der sich wiederholt, mit einem Buchstabenkommando (z.B. „N") an und endet mit 34: n, wobei n die hexadezimale Position des Zeichens ist, an dem die Schleife anfängt (für das erste Zeichen ist n = 0). Nach dem Wert für n muß ein Leerzeichen folgen, damit diese Schleife funktioniert.

```
*N 300 302 34:0

0300- 11
0302- 33
0300- 11
0302- 33
0300- 11
0302- 33
0300- 11
0302- 33
0300- 11
0302- 33
0300- 11
0302- 33
030
*
```

Solch eine Schleife läßt sich nur durch ein RESET stoppen.

Erzeugen eigener Kommandos

Das USER-Kommando wird durch ein CTRL Y in einer Eingabe gegeben und läßt den Monitor zur Adresse $3F8 springen. Sie können in diese Adresse einen JMP-Befehl einsetzen, der zu Ihrem eigenen Programm springt. Ihr Programm kann die Register, die Spezialadressen des Monitors oder die Eingabezeile prüfen. Beispielsweise kann das CTRL Y-Kommando zu einem „Kommentar"-Kommando programmiert werden: In einer Eingabezeile wird alles dem CTRL Y folgende angezeigt und ignoriert.

```
*F666G

!300:LDY $34

0300-     A4 34      LDY    $34
! LDA 200,Y

0302-     B9 00 02   LDA    $0200,Y
! JSR FDED

0305-     20 ED FD   JSR    $FDED
! INY

0308-     C8         INY
! CMP $8D

0309-     C9 8D      CMP    #$8D
! BNE 302

030B-     D0 F5      BNE    $0302
! JMP $FF69

030D-     4C 69 FF   JMP    $FF69
!3F8:JMP $300

03F8-     4C 00 03   JMP    $0300

!$FF69G

*CTRL Y DIES IST EIN TEST

DIES IST EIN TEST

*
```

Übersicht über die Monitor-Kommandos

Speicherzellen ansehen

{Adresse}	Gibt den Inhalt einer Speicherstelle aus.
{Anfang} . {Ende}	Gibt alle Inhalte zwischen {Anfang} und {Ende} aus.
RETURN	Zeigt die Werte von maximal acht Speicherstellen nach der zuletzt geöffneten Adresse an.

Speicherinhalte verändern

{Adresse} : {Wert} {Wert}...	Speichert ab {Adresse} die Werte in aufeinanderfolgende Speicherzellen.
: {Wert} {Wert}...	Speichert ab der nächsten veränderbaren Adresse die Werte in aufeinanderfolgende Speicherzellen.

Transportieren und Vergleichen

{Ziel} < {Anfang} . {Ende} M	Kopiert die Werte des Bereichs {Anfang} . {Ende} in den Bereich ab {Ziel}.
{Ziel} < {Anfang} . {Ende} V	Vergleicht die Werte des Bereichs {Anfang} . {Ende} mit dem Bereich ab {Ziel}.

Schreiben und Lesen auf Band

{Anfang} . {Ende} W	Schreibt die Werte des Bereichs nach einer 10s-Kopfinformation auf Band.
{Anfang} . {Ende} R	Liest Werte vom Band in den Speicherbereich {Anfang} . {Ende}. Druckt „ ERR " im Fehlerfall.

Starten und Ausdrucken von Programmen

{Adresse} G	Läßt den Mikroprozessor ab {Adresse} das Maschinenprogramm ausführen.
{Adresse} L	Läßt ab {Adresse} 20 Befehle des Maschinenprogramms in Assemblersprache (Disassembler) ausgeben. Jedes weitere „ L" läßt 20 weitere Befehle ausgeben.

Der Mini-Assembler

F666G	Startet den Mini-Assembler*.
$ {Kommando}	Führt ein Monitor-Kommando aus.
$FF69G	Beendet den Mini-Assembler und führt in den Monitor.
{Adresse} S	Zeigt den Maschinenbefehl von Adresse in Assemblersprache an, führt diesen Befehl aus und zeigt die internen Register des 65Ø2 an. Jedes nachfolgende „S" zeigt den nächsten Befehl an und führt ihn aus**.
{Adresse} T	Dauerndes S-Kommando. Wird nur durch Erreichen eines BRK-Befehls oder ein RESET gestoppt**.
CTRL E	Zeigt die Inhalte der 65Ø2-Register an.

Verschiedenes

I	Setzt INVERSE-Modus.
N	Setzt NORMAL-Modus.
CTRL B	Startet die Sprache, die im ROM des Apple verfügbar ist.
CTRL C	Setzt die Sprache fort, die im ROM des Apple verfügbar ist.
{Wert} + {Wert}	Addiert zwei Werte und druckt das Ergebnis.
{Wert} – {Wert}	Subtrahiert den zweiten Wert vom ersten und zeigt das Ergebnis an.
{Nummer} CTRL P	Bestimmt die Ausgabe zu dem Gerät, dessen Interface-Karte im Geräteschlitz {Nummer} steckt. Ist {Nummer}=Ø, dann kommt die Ausgabe auf den Bildschirm.
{Nummer} CTRL K	Nimmt die Eingabe von dem Gerät an, dessen Interface-Karte im Geräteschlitz {Nummer} steckt. Ist {Nummer}=Ø, dann wird die Eingabe von der Tastatur erwartet.
CTRL Y	Springt zu dem Maschinen-Unterprogramm ab Adresse $3F8.

* Nicht verfügbar im Apple II Plus.
** Nicht verfügbar im Autostart-ROM.

Einige nützliche Monitor-Unterprogramme

Diese Liste enthält einige nützliche Unterprogramme im Monitor-ROM und Autostart-ROM des Apple. Vor dem Aufruf der Unterprogramme laden Sie die nötigen Speicheradressen oder 6502-Registerinhalte. Der Aufruf erfolgt durch einen JSR-Befehl (Sprung ins Unterprogramm) zu der angegebenen Startadresse des Unterprogramms. Es wird die beschriebene Funktion ausführen und die Register so hinterlassen, wie es jeweils angegeben ist. Der Prozessorstatus (C, Z, N, V) wird im allgemeinen geändert.

$FDED　　COUT　　Ausgabe eines Zeichens (Character OUTput)

COUT ist das Standard-Unterprogramm für Zeichenausgabe. Das Zeichen, das ausgegeben werden soll, steht im Akkumulator. COUT ruft das aktuelle Zeichenausgabe-Unterprogramm auf, dessen Adresse in CSW (Adressen $36 und $37) steht. Normalerweise ist es COUT 1.

$FDF0　　COUT1　　Ausgabe auf den Bildschirm

COUT 1 bringt das Zeichen im Akkumulator auf den Bildschirm des Apple. Es wird auf die laufende Position des Ausgabezeigers gesetzt und bewegt dann diesen Zeiger weiter. Das Zeichen wird mit dem Inhalt der NORMAL-/INVERSE-Speicherstelle modifiziert. Die Steuerzeichen RETURN, Zeilenvorschub und Klingelzeichen werden von COUT 1 ebenfalls behandelt. Das Unterprogramm läßt alle Register intakt.

$FE80　　SETINV　　Setzt den INVERSE-Modus (SET INVerse)

Der INVERSE-Modus für COUT 1 wird gesetzt. Dadurch erscheinen alle Zeichen als schwarze Punkte auf weißem Hintergrund, die dann von COUT 1 ausgegeben werden. Das Y-Register wird auf $3F gesetzt, alle anderen Register bleiben unverändert.

$FE84　　SETNORM　　Setzt den NORMAL-Modus (SET NORMal)

Setzt den NORMAL-Modus für COUT 1. So werden alle Zeichen als weiße Punkte auf schwarzem Hintergrund ausgegeben. Das Y-Register erhält den Wert $FF, alle anderen Register bleiben unverändert.

$FD8E　　CROUT　　Gibt ein RETURN aus (Carriage Return OUTput)

CROUT sendet ein RETURN zu dem aktuellen Ausgabegerät.

$FD8B　　CROUT1　　RETURN mit Löschfunktion

CROUT 1 löscht die Zeile auf dem Bildschirm von der position des Ausgabezeigers bis zur rechten Begrenzung des Textfensters. Dann wird CROUT aufgerufen.

$FDDA　　PRBYTE　　Druckt ein Byte als Hexadezimalzahl

Dieses Unterprogramm gibt den Inhalt des Akkumulators als Hexadezimalzahl auf das aktuelle Ausgabegerät. Der Inhalt des Akkumulators wird verändert

$FDE3　　PRHEX　　Druckt eine Hexadezimalziffer (PRint HEXadecimal digit)

Dieses Unterprogramm gibt die unteren vier Bits (Bit 3 bis Bit 0) des Akkumulators als eine Hexadezimalziffer aus. Der Inhalt des Akkumulators wird verändert.

$F941 PRNTAX Druckt A und X als eine Hexadezimalzahl (PRiNT A and X in hexadecimal)

Dieses Unterprogramm gibt die Inhalte des Akkumulators und des X-Registers als vierziffrige Hexadezimalzahl aus. Der Akkumulator enthält die linken zwei Ziffern, das X-Register bestimmt die rechten zwei Ziffern. Der Inhalt des Akkumulators wird verändert.

$F948 PRBLNK Druckt drei Leerzeichen (PRint 3 BlaNK spaces)

Gibt drei Leerzeichen über das Standard-Ausgabegerät aus. Der Akkumulator bekommt den Wert $A0 und das X-Register den Wert 0.

$F94A PRBL2 Druckt viele Leerzeichen

Dieses Unterprogramm gibt 1 bis 256 Leerzeichen zur Standardausgabe. Beim Aufruf bestimmt der Inhalt des X-Registers die Anzahl der Leerzeichen. Ist X =0, so werden 256 Leerzeichen ausgegeben. Beim Ausgang hat der Akkumulator den Inhalt $A0 und das X-Register den Inhalt 0.

$FF3A BELL Ausgabe eines Klingel-Zeichens (BELL)

Dieses Unterprogramm sendet ein Klingel-Zeichen (CTRL G) zu dem aktuellen Ausgabegerät. Der Akkumulator bekommt den Wert $87.

$FBDD BELL1 Abgabe eines Tonsignals aus dem Lautsprecher des Apple

Dieses Unterprogramm erzeugt für die Dauer einer Zehntelsekunde einen Ton von 1 KHz. Die Inhalte des Akkumulators und des X-Registers werden verändert.

$FD0C RDKEY Eingabe eines einzelnen Zeichens

Dies ist das Unterprogramm für Standard-Zeicheneingabe. Ein blinkender Eingabezeiger erscheint auf dem Bildschirm an der Position des Ausgabezeigers und das Unterprogramm springt zu dem aktuellen Eingabe-Unterprogramm, dessen Adresse in KSW (Adressen $38 und $39), gewöhnlich KEYIN (siehe unten), zu finden ist.

$FD35 RDCHAR Eingabe eines einzelnen Zeichens oder einer ESC-Anweisung

RDCHAR ist ein weiteres Eingabe-Unterprogramm, das Zeichen von der Standardeingabe erhält, aber auch die elf Escape-Anweisungen interpretiert (siehe Seite 43).

$FD1B KEYIN Lesen eines Zeichens von der Tastatur

Dies ist das Unterprogramm für die Eingabe über die Tastatur. Die Apple-Tastatur wird abgefragt, es wird auf einen Tastendruck gewartet und eine Zufallszahl gebildet (siehe Seite 40). Erfolgt ein Tastendruck, so wird der blinkende Zeiger entfernt und der Tastencode in den Akkumulator gegeben.

$FD6A GETLN Anforderung einer Eingabezeile mit Bereitschaftszeichen

Das Unterprogramm GETLN sammelt aus einzelnen Zeichen eine Eingabezeile (siehe Seite 41). Ihre Programme können das Bereitschaftszeichen für GETLN in der Speicherzelle $33 bestimmen. Das Unterprogramm GETLN kerht mit der Eingabezeile im Eingabepuffer (ab Adresse $200) und mit der Länge der Eingabezeile im X-Register zurück.

$FD67 GETLNZ Anforderung einer Eingabezeile

Das Unterprogramm GETLNZ schickt erst einen Zeilenvorschub zum Standardausgabegerät, bevor GETLN ausgeführt wird (siehe oben).

$FD6F GETLN1 Anforderung einer Eingabezeile ohne Bereitschaftszeichen

GETLN1 beginnt in GETLN erst an der Stelle, an der die Eingabezeile gebildet wird, so daß kein Bereitschaftszeichen erscheint. Löschen Sie jedoch mehr Zeichen als in der Eingabezeile vorhanden waren oder betätigen Sie CTRL X, so gibt GETLN1 den Inhalt der Speicherzelle $33 als Bereitschaftszeichen einer neuen Eingabezeile aus.

$FCA8 WAIT Warten

Dieses Unterprogramm wartet eine bestimmte Zeit und kehrt dann wieder zu dem Programm zurück, das es aufgerufen hat. Der Akkumulator bestimmt diese Zeit. Wenn A der Inhalt des Akkumulators ist, ergibt sich eine Verzögerung von $(26 + 27A + 5A^2)/2$ Mikrosekunden (13 Mikrosekunden bis ca. 1/6 Sekunde). WAIT läßt X und Y unverändert, nur das A-Register wird Ø.

$F864 SETCOL Setzt die Farbe für die Ausgabe von Lo-Res Graphik (SET COLor)

Der Akkumulator bestimmt die Farbe, die bei der Lo-Res Graphik-Ausgabe auf den Bildschirm verwendet werden soll. Die Farben finden Sie in Tabelle 8 auf Seite 21. Der Akkumulator wird verändert, sonst ändern sich die Register nicht.

$F85F NEXTCOL Die Farbnummer wird um 3 erhöht (NEXT COLor)

Die aktuelle Farbe für die Ausgabe von Lo-Res Graphik wird um 3 erhöht. Nur das A-Register wird verändert.

$F8ØØ PLOT Druckt einen Block auf den Lo-Res Bildschirm

Dieses Unterprogramm druckt einen einzelnen Block in der vorher eingestellten Farbe auf den Bildschirm. Die vertikale Position wird im Akkumulator übergeben und die horizontale Position wird dem Y-Register entnommen. PLOT verändert nur den Akkumulator.

$F819 HLINE Zeichnet eine waagrechte Linie von Blöcken

Es wird eine Zeile von Blöcken in der vorher festgelegten Farbe auf den Lo-Res-Bildschirm gezeichnet. Folgende Angaben müssen beim Aufruf vorhanden sein: Die senkrechte Koordinate steht im Akkumulator, die waagrechte Koordinate des linken Endes im Y-Register, die des rechten Endes in $2C. HLINE verändert A und Y, läßt aber X intakt.

$F828 VLINE Zeichnet eine senkrechte Linie von Blöcken

Dieses Unterprogramm zeichnet eine senkrechte Linie von Blöcken der vorher festgelegten Farbe auf den Lo-Res-Bildschirm. Folgende Werte müssen beim Aufruf vorliegen: Die oberste vertikale Position im Akkumulator, die unterste vertikale Koordinate in $2D und die horizontale Koordinate der Linie im Y-Register. VLINE verändert den Akkumulator.

$F832 CLRSCR Löscht den gesamten Lo-Res Bildschirm

CLRSCR löscht den gesamten Bildschirm der Blockgraphik. Wird CLRSCR im TEXT-Modus aufgerufen, so wird der Bildschirm mit inversen „@"-Zeichen gefüllt. CLRSCR verändert die Inhalte von A und X.

$F836 CLRTOP Löscht den oberen Teil der Lo-Res Graphik

CLRTOP arbeitet wie CLRSCR (siehe oben), aber es werden nur die oberen 40 Reihen des Bildschirms gelöscht.

$F871 SCRN Liest ein Zeichen auf dem Lo-Res Bildschirm

Dieses Unterprogramm kehrt mit der Farbe eines bestimmten Blocks auf dem Bildschirm in das Programm zurück, das SCRN aufgerufen hat. Den Aufruf gestalten Sie wie bei PLOT (siehe oben). Die Nummer der Farbe des Blocks steht nach dem Aufruf im Akkumulator. Andere Register werden nicht verändert.

$FB1E PREAD Liest die Stellung einer Spielsteuerung

PREAD braucht zum Aufruf die Nummer der Spielsteuerung im X-Register. Diese Zahl muß Ø, 1, 2 oder 3 sein, sonst werden Sie sich wundern. Die Stellung der Spielsteuerung wird als Zahl zwischen $ØØ und $FF im Y-Register übergeben. Der Akkumulator wird verändert.

$FF2D PRERR Drucke „ERR"

PRERR sendet die Zeichen „E", „R", „R" und das Klingelzeichen zum Standardausgabegerät. Der Akkumulator wird verändert.

$FF4A IOSAVE Rettet alle Register

Die Inhalte aller internen Register des 65Ø2-Mikroprozessors werden in der Reihenfolge A-X-Y-P-S in die Speicherstellen $45 bis $49 geschrieben. Die Inhalte von A und X werden verändert und der Dezimalmodus des Mikroprozessors wird gelöscht.

$FF3F IOREST Alle Register werden wiederhergestellt

Die Inhalte der internen Register des 65Ø2-Mikroprozessors werden von den Speicherstellen $45 bis $49 geladen.

Spezialadressen des Monitors

Tabelle 14: Monitor-Adressen auf Seite 3

Adresse Dezimal	Hexa	Verwendung Monitor-ROM	Autostart-ROM
1008 1009	$3F0 $3F1	keine	Enthält die Adresse des Unterprogramms, das "BRK"-Nachfragen behandelt (normal: $FA59).
1010 1011	$3F2 $3F3	keine	Eingangsadresse in die benutzte Sprache.
1012	$3F4	keine	Einschalt-Byte
1013 1014 1015	$3F5 $3F6 $3F7	Enthält einen JMP (Sprung)-Befehl zu dem Unterprogramm, das Applesoft II "&"-Kommandos behandelt*. (Normal: $4C $58 $FF)	
1016 1017 1018	$3F8 $3F9 $3FA	Enthält einen JMP-Befehl zu dem Unterprogramm, das "USER" (CTRL Y) -Kommandos behandelt.	
1019 1020 1021	$3FB $3FC $3FD	Enthält einen JMP-Befehl zu dem Unterprogramm, das nichtmaskierbare Interrupts behandelt.	
1022 1023	$3FE $3FF	Enthält die Adresse des Unterprogramms, das Interrupt-Nachfragen (IRQ) behandelt.	

Format der Mini-Assembler Befehle

Der Apple Mini-Assembler erkennt 56 Befehlssymbole und 13 Adreßarten, die in der 6502 Assembler-Programmierung verwendet werden. Die Befehlssymbole entsprechen dem Standard, der im „MOS Technology/Synertek 6500 Programming Manual" (Apple Teil A2L0003) angegeben ist, aber die Adreßformate sind dort anders. Hier sind die Standard-Formate der 6502-Adreßarten :

* Siehe Seite 123 des Applesoft BASIC Handbuchs.

Tabelle 15: Mini-Assembler Adreßformate	
Adreßart:	Format:
Akkumulator	keine Adresse
Unmittelbar	#$ {Wert}
Absolut	$ {Adresse}
Seite ∅	$ {Adresse}
Seite ∅ indiziert	$ {Adresse} ,X $ {Adresse} ,Y
Absolut indiziert	$ {Adresse} ,X $ {Adresse} ,Y
Impliziert	keine Adresse
Relativ	$ {Adresse}
Indiziert indirekt	($ {Adresse} ,X)
Indirekt indiziert	($ {Adresse}),Y
Absolut indirekt	($ {Adresse})

Eine Adresse besteht aus mindestens einer Hexadezimalziffer. Der Mini-Assembler wertet die Adressen wie der Monitor aus : hat eine Adresse weniger als vier Ziffern, so bekommt sie führende Nullen; hat sie mehr als vier Ziffern, so werden nur die vier rechten Ziffern ausgewertet.

Alle Dollar-Zeichen ($), die die hexadezimale Schreibweise anzeigen, werden vom Mini-Assembler ignoriert und können weggelassen werden.

Es gibt keine syntaktischen Unterschiede zwischen absoluter Adressierung und Adressierung in Seite 0. Geben Sie einen Befehl, der auf diese beiden Weisen adressiert werden kann, so wird er in einen Befehl mit absoluter Adressierung übersetzt, falls der Operand dieses Befehls größer als $FF ist. Ist der Operand kleiner als $0100, so wird es ein Befehl für Seite 0-Adressierung.

Befehle mit der Akkumulator- oder implizierten Adressierung brauchen keinen Operanden.

Für Verzweigungs-Befehle (Branch), die „relativ" adressieren, sind die Zieladressen als Adreßangabe erforderlich. Der Mini-Assembler rechnet automatisch die relative Sprungweite aus, die eigentlich für den Befehl nötig ist. Ist die Zieladresse mehr als 127 Speicherstellen von dem Befehl entfernt, so generiert der Mini-Assembler einen Ton, setzt einen Zirkumflex (^) unter die Zieladresse und ignoriert die Eingabezeile.

Geben Sie dem Mini-Assembler ein Befehlssymbol und einen Operanden, dessen Adressierungsart nicht zu dem Befehl paßt, wird der Mini-Assembler diese Eingabe nicht annehmen.